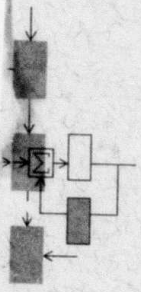
LEVEL II

12

November, 1981

LIDS-TH-1162

Research Supported By:
ARPA Contract N00014-75-C-1183



MULTIPLE ACCESS COMMUNICATION: THE FINITE USER POPULATION PROBLEM

Michael Gene Hluchyj

Laboratory for Information and Decision Systems
MASSACHUSETTS INSTITUTE OF TECHNOLOGY, CAMBRIDGE, MASSACHUSETTS 02139

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A108337	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Multiple Access Communication: The Finite User Population Problem		5. TYPE OF REPORT & PERIOD COVERED Thesis
		6. PERFORMING ORG. REPORT NUMBER LIDS-TH-1162
7. AUTHOR(s) Michael Gene Hluchyj		8. CONTRACT OR GRANT NUMBER(s) ARPA Order No. 3045/5-7-75 ONR/N00014-75-C-1183
9. PERFORMING ORGANIZATION NAME AND ADDRESS Massachusetts Institute of Technology Laboratory for Information and Decision Systems Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Program Code No. 5T10 ONR/N00014-75-C-1183
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE November, 1981
		13. NUMBER OF PAGES 143 pages 146
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Program Code 437 Arlington, Virginia 22217		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The multiple access problem is one of organizing a population of users so that they may efficiently share the resources of a single communication channel. This problem is examined under the modeling assumptions of a finite user population and a time-slotted channel with limited feedback.

Techniques or schemes for coordinating the transmissions of users are called multiaccess protocols. Simple relationships among common steady-state measures of protocol performance (including throughput and average delay) are derived. From these relationships it is shown that the performance measures are

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

equivalent in the sense that (1) each may be expressed as a simple function of any one of the others and (2) a protocol which is optimal with respect to any one measure is optimal with respect to the others. The derived relationships are also used in the performance analysis of perfect scheduling and TDMA.

In the area of protocol development, four related classes of multiaccess protocols are defined and examined. The most general class of protocols is considered first, and the other three are subclasses of it. For each class the problem of finding an optimal protocol is characterized. The optimization problem is formulated as a Team problem for the first class, and as a Markov decision problem for each of the other three classes. However, only with the last class examined, the Window protocols, does the optimization problem prove to be tractable.

Using results from Markov decision theory, optimal Window protocols are derived for the cases of two and three users. The Window protocol state space, however, grows exponentially with the population size and this prevents an exact determination of optimal protocols for large user populations. For this case, a Window protocol subclass is defined and an approximate analysis is used to determine the performance and dynamic behavior of protocols within this subclass. Also, with this analysis a link is established between the finite and infinite population problems.

November, 1981

LIDS-TH-1162

MULTIPLE ACCESS COMMUNICATION: THE FINITE USER POPULATION PROBLEM

by

Michael Gene Hluchyj

This report is based on the unaltered thesis of Michael Gene Hluchyj, submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology in November, 1981. The research was conducted at the M.I.T. Laboratory for Information and Decision Systems, with support provided in part by the Advanced Research Project Agency under Contract No. N00014-75-C-1183.

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

MULTIPLE ACCESS COMMUNICATION:

THE FINITE USER POPULATION PROBLEM

by

Michael Gene Hluchyj

B.S., University of Massachusetts at Amherst
(1976)

S.M., Massachusetts Institute of Technology
(1978)

E.E., Massachusetts Institute of Technology
(1978)

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

November, 1981

Signature of Author ..... Michael G Hluchyj .....
Department of Electrical Engineering and Computer Science
November 6, 1981

Certified by ..... Robert G. Gallager .....
Thesis Supervisor

Accepted by .....
Chairman, Departmental Committee on Graduate Students

MULTIPLE ACCESS COMMUNICATION:

THE FINITE USER POPULATION PROBLEM

by

Michael Gene Hluchyj

Submitted to the Department of Electrical Engineering and Computer Science on November 6, 1981, in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

ABSTRACT

The multiple access problem is one of organizing a population of users so that they may efficiently share the resources of a single communication channel. This problem is examined under the modeling assumptions of a finite user population and a time-slotted channel with limited feedback.

Techniques or schemes for coordinating the transmissions of users are called multiaccess protocols. Simple relationships among common steady-state measures of protocol performance (including throughput and average delay) are derived. From these relationships it is shown that the performance measures are equivalent in the sense that (1) each may be expressed as a simple function of any one of the others and (2) a protocol which is optimal with respect to any one measure is optimal with respect to the others. The derived relationships are also used in the performance analysis of perfect scheduling and TDMA.

In the area of protocol development, four related classes of multiaccess protocols are defined and examined. The most general class of protocols is considered first, and the other three are subclasses of it. For each class the problem of finding an optimal protocol is characterized. The optimization problem is formulated as a Team problem for the first class, and as a Markov decision problem for each of the other three classes. However, only with the last class examined, the Window protocols, does the optimization problem prove to be tractable.

Using results from Markov decision theory, optimal Window protocols are derived for the cases of two and three users. The Window protocol state space, however, grows exponentially with the population size and this prevents an exact determination of optimal protocols for large user populations. For this case, a Window protocol subclass is defined and an approximate analysis is used to determine the performance and dynamic behavior of protocols within this subclass. Also, with this analysis a link is established between the finite and infinite population problems.

Thesis Supervisor: Robert G. Gallager
Title: Professor of Electrical Engineering

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Professor Robert G. Gallager, my thesis supervisor, for his invaluable advice and support. His keen insights and observations, coupled with his personable and flexible manner, have made this research experience both rewarding and enjoyable.

I also wish to express my appreciation to the readers of this thesis, Professors Pierre A. Humblet and Dimitri P. Bertsekas, for the helpful comments which they have provided. In addition, during my stay at MIT I have benefited from numerous discussions with faculty members and fellow graduate students. In specific relation to this thesis, I thank Professor Charles Van Loan who introduced me to the numerical method presented in Appendix A, and Mr. Eli Gafni who suggested a simplification for the proof of Theorem 1 in Appendix B.

Lastly, and most importantly, I wish to thank Terry, my wife, not only for her technical assistance in producing this report, but also for her love and support throughout my graduate career.

This work was supported by a Research Assistantship from the Laboratory for Information and Decision Systems with funds provided by DARPA Contract ONR/N00014-75-C-1183.

TABLE OF CONTENTS

TABLE OF CONTENTS (con't)

TABLE OF CONTENTS (con't)

LIST OF FIGURES

LIST OF FIGURES (con't)

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Multiple Access Problem

A communication problem that has received much attention during the past decade is that of organizing a population of users (also called sources) so that they may efficiently share the resources of a single communication channel. Although various models of the user population and communication channel have been considered, they generally have the following properties. The users are geographically distributed and generate messages (i.e., blocks of digital data to be transported over the channel) in an independent random fashion. The channel is such that only one user at a time can successfully transmit a message, and associated with message transmissions is some form of feedback to the users. This feedback has typically ranged from no feedback (e.g., TDMA [M3]) to each individual user determining whether its own message transmissions are successful (e.g., Aloha [A1]) to every user determining after some given delay whether there are 0, 1, or ≥ 2 messages being transmitted on the channel (e.g., Ethernet [M4], Tree [C2]).

The problem of organizing or coordinating the transmissions of users for the efficient utilization of the channel is referred to as the multiple access (or multiaccess) problem, and arises most often in the context of a broadcast communication system. Here each message generated by a user (e.g., terminal, computer, sensor) is to be transported via a common broadcast channel to one or more of the other users. Although it varies with the particular system model, it is generally the broadcast nature of the channel that provides the feedback of information concerning message

transmissions to the user population. Examples of such channels include a satellite channel where the satellite acts as a transponder, a ground radio channel with all users within transmission range of one another, and a length of coaxial or fiber optic cable to which the users are physically attached. From these examples it is clear that "geographically distributed" may imply anything from users located in an area of several million square kilometers to users located in a small room.

The advantages of having a user population share a common broadcast channel have been well documented in the literature (see, e.g., [A1,K3]). Briefly, the desirable aspects of such a system include (1) the possibility of utilizing efficiently a communications resource in an environment of many bursty users, (2) the high connectivity of the system where any user can directly communicate with any other user, (3) the broadcast nature of the channel which allows a single transmitted message to be delivered to many users, and (4) the inherent flexibility associated with adding and removing users and, in the case of a radio channel, with moving the users without physically reconfiguring the system.

Although later we will be more precise, the efficient utilization of the communication channel basically involves avoiding two undesirable events. One is referred to as a collision and is when two or more message transmissions overlap in time, thus destroying each other so that none are received successfully. The other is when nothing is being transmitted (the channel is idle) and there is at least one user with one or more buffered messages awaiting transmission (a system busy period). Both of these events correspond to the channel being wasted in the sense that there are one or more messages waiting to be "serviced" (i.e., successfully transmitted) and none are actually getting service. Unfortunately,

avoiding both events is generally impossible.

The random generation of messages coupled with the distributed nature of the users precludes any so called "perfect scheduling" of the messages where both collisions and channel idles during busy periods are eliminated. If the message generation times are random but the users are not distributed, then perfect scheduling may be accomplished by forming a common queue of the generated messages and then servicing them, for example, on a first-come first-serve basis. Likewise, if the users are distributed but the message generation times and block sizes are deterministic, then the use of the channel could clearly be (perfectly) scheduled beforehand to avoid both collisions and channel idles during busy periods. With both distributed users and randomly generated messages, the information needed to implement perfect scheduling is dispersed among the users and the only means to exchange this information is through the channel they wish to access in the first place.

The techniques or schemes for coordinating the transmissions of the users are called multiaccess protocols. Later we shall see that avoiding collisions and avoiding channel idles during busy periods are conflicting goals, and that designing an efficient multiaccess protocol essentially involves trading off these two undesirable events in such a way as to achieve the best possible system performance.

1.2 The Selected User-Channel Model

In this section we state an explicit model for the user population and communication channel. All analytical work that follows is based on this given model. In the next section we describe some of the other modeling assumptions that have been chosen in the past and indicate why we have made

the given selection.

We consider a finite population of N users, where the messages generated by each user are incorporated into fixed-length blocks of data called packets. Each packet, in addition to the message data, generally contains control information such as source and destination addresses. Associated with the fixed-length packets is a time-slotted channel. By this we mean that packet transmissions are synchronized to occur within globally defined time-slots, where the slot size is equal to the time to transmit one packet. Hence one can envision the channel as a succession of rectangular slots into which the users transmit their packets.

It is assumed that a given slot results in a successful packet transmission if and only if the slot contains exactly one packet. A slot occupied by two or more packets results in a collision where none are successful, requiring each to be retransmitted at a later time. When no packet transmission occurs within a slot, we say the slot is empty. As for the channel feedback, immediately following the end of each slot, it is assumed that each user can determine whether the slot contained 0, 1, or ≥ 2 packets, corresponding to, respectively, an empty slot, a success, or a collision. With a typical broadcast channel, the distinction between a success and a collision may be accomplished with the use of error detection information included in the packet (e.g., a cyclic-redundancy-check), and an empty slot distinguished from the other two outcomes through the absence of signal energy on the channel.

Finally, we assume a homogeneous population of users, where at the beginning of each slot each user which does not currently have a packet awaiting transmission will independently generate one with probability p. This is equivalent to the "single buffer" assumption where at the beginning

of each slot each user independently generates a packet with probability p, but will only accept this packet into its transmission buffer if the buffer is currently empty. Note that the buffer does not actually become empty until the end of the slot during which the stored packet is successfully transmitted (i.e., an unsuccessfully transmitted packet remains in the buffer). Any packets that are generated while the buffer is not empty are assumed lost. Also, a packet entering the transmission buffer at the beginning of a slot may be transmitted in that slot.

1.3 Other Possible Modeling Assumptions

The modeling assumptions given in the previous section are quite common in the study of the multiaccess problem. However, they are by no means unique. In this section we briefly describe and discuss some of the other popular modeling assumptions. We shall see that many of these assumptions are, conceptually, just simple extensions to the model we have selected.

We begin with the carrier sense assumption. First used in the development of multiaccess protocols by Kleinrock and Tobagi [K4], the concept behind carrier sensing is very simple: each user is able to detect, after some given delay, when the channel switches between being used (carrier present) and idle (carrier absent). In an unslotted channel, particularly one that permits variable-length packets, carrier sense has been used to reduce the rate of collisions by having users refrain from transmitting packets when a carrier is detected. For a slotted channel with fixed-length packets, the usefulness of carrier sense is in allowing all users to determine within some given delay following the start of each slot whether that slot contains 0 or ≥ 1 packets. Typically this delay

consists of the channel propagation delay. Hence for a channel whose propagation delay is smaller than the chosen slot size (often true for ground radio and cable systems), one can, with carrier sense, define empty slots to be smaller than slots containing successes or collisions. In this way the "cost" associated with an empty slot during a busy period is reduced, resulting in the potential for increased efficiency.

Further performance improvements can be realized with the addition of the collision detection assumption. First introduced by Metcalfe and Boggs [M4], collision detection refers to the ability of all users to detect a collision before the transmissions are completed, thus allowing the interfering transmissions to be aborted. This assumption is valid for channels with a small propagation delay and a listen-while-transmit feature (e.g., some cable systems). Combined with carrier sense, it results in collision and empty slots smaller than slots with successes.

Relevant to this discussion is the concept of packet reservation. Brought into prominence by Roberts [R3], the typical approach with packet reservation is for users to inform one another of generated messages by transmitting over the broadcast channel smaller "reservation" packets containing this information. Once all users are aware that a particular user has a message packet awaiting transmission, they can (by following a common algorithm) collectively reserve a future time-slot for which this user will have exclusive transmission rights. The multiaccess problem is still present, but now it is associated with the reservation process. However, since these reservation packets are smaller in size, their use can often improve the overall system performance. In any event, we see that the reservation problem may be viewed as an extension of the basic multiaccess problem. This is particularly apparent when one considers, for

E - empty
S - success
C - collision

Figure 1-1 Channel usage by a reservation scheme

a small propagation delay channel, the reservation scheme whereby a message packet is transmitted immediately following the successful reception of its corresponding reservation packet. This process is illustrated in Figure 1-1. Note that by viewing the successful transmission as consisting of both the message and its reservation, we again have a system whose slot size varies according to the channel outcome: collision and empty slots are smaller than those with successes.*

Having a buffer size >> 1 at each user to queue generated messages before transmission may at first seem more acceptable than the selected single buffer assumption. With a larger buffer one can generally reduce the probability that a generated message is blocked from entering the buffer. Also, there is more flexibility in modeling the arrival process. For example, a multiple packet generation process (i.e., bulk arrivals to the buffer) may be used to model variable-length messages, the length being some varying multiple of the slot size. The single buffer assumption is analytically more tractable and in past work it was justified by

* Within the context of an infinite population model, Humblet [H7,H8] has generalized several results, originally derived for the case of fixed-length slots, to the case where the slot size is a fixed but arbitrary function of the channel outcome $\{0,1,\geq 2\}$.

considering a very large (often infinite) population of users with individual packet generation rates sufficiently small that a packet arriving at the buffer would very likely find it empty. We, however, are interested in all finite populations, starting with $N = 2$, and all packet generation probabilities, $0 < p < 1$.

Our justification for the single buffer assumption, as with many of the other modeling assumptions, is that it in some sense represents the basic multiaccess problem. As indicated earlier, the essential difficulty in dealing with the multiaccess situation is derived from the lack of common knowledge as to which users have packets awaiting transmission. With the buffer size larger than one, this becomes less of a problem in a broadcast environment since the control portion of a user's successfully transmitted packet may be used to inform the other users of any additional packets remaining in its buffer. This may be done in a variety of ways. One particularly simple method is to use a one bit field in the packet header to indicate the presence/absence of other buffered packets at the instant before transmission. With this information it follows that each packet generated during a user's busy period (i.e., the continuous interval during which the user's buffer is not empty) may be assigned a reserved slot for transmission. The multiaccess problem is then only associated with the first packet at the start of each of these busy periods. In general, a higher packet generation probability p implies a longer user busy period and hence a smaller fraction of packets without reserved slots. It is apparent then that with a buffer size larger than one, the uncertainty as to which users have packets awaiting transmission is reduced. Since this uncertainty is at the heart of the multiaccess problem, the single buffer assumption represents the problem in its most

basic form.

The use of an infinite user population model originally came about as an approximation to a large but finite population in the analysis of a particular multiaccess protocol [K2]. Since then a significant amount of work in the multiaccess area has centered around this assumption. Of primary interest to many of these studies is the "capacity" of the multiaccess channel, defined as the supremum of all achievable throughputs for which the average packet delay is finite. Part of this effort has been devoted toward devising multiaccess protocols with greater throughput [C1,G1,M8], and part devoted to determining tighter theoretical upper bounds on the capacity [P2,H6,M7,C6,T3]. Currently, for a channel with $\{0,1,\geq 2\}$ feedback, the largest throughput attained by a protocol is 0.4877 [M8] and the tightest upper bound now stands at 0.5874 [T3]. Unfortunately, these results have only limited application to the more practical case of a finite user population; for the maximum throughput of the finite population problem is 1.0 and is achieved by TDMA (Time-Division Multiple Access) where slots are assigned to users in a deterministic round-robin fashion (see Section 2.3). Hence caution must be exercised in using the more tractable infinite population model as an approximation to a large but finite population, as the two clearly have significant fundamental differences.

The different modeling assumptions given in this section are only but a sample of the possible variations that one could consider. By relaxing the basic properties given in Section 1.1, we could even enlarge the scope of the multiaccess problem to include such models as those used in the study of packet radio networks [K1]. However, we believe that to understand the fundamental nature of multiaccess, one should first examine

the basic problem. The model of Section 1.2 represents such a basic problem.

1.4 Previous Work

There has been a great deal of effort devoted to the multiple access problem and much of this work has followed along the lines of protocol development and analysis. The numerous protocols found in the literature reflect not only the fact that there are many different models of the user population and communication channel (each possibly motivated by a different practical situation), but also that the problem of finding the best protocol for any nontrival model and performance measure is as yet unsolved. In this section we briefly describe some of the main contributions in the development of multiaccess protocols. The attempt is to give an overview of the various classes of protocols, particularly those relevant to our selected model, without getting sidetracked with the details of any one scheme. Other more complete surveys may be found in the literature [T1,L2].

The first distinction we make is between protocols that allow conflicts and those that are conflict-free. As the name implies, a conflict-free protocol requires that no more than one user transmit at any given time. Not surprisingly, the first protocols developed for the multiaccess problem were conflict-free. One important such protocol is TDMA (Time-Division Multiple Access) [M3]. Although variations do exist, the typical TDMA protocol consists of assigning fixed, predetermined slots to users in a round-robin fashion. In this way, with N users, each user is given periodic access to the channel once every N slots. Without conflicts, the performance degradation of TDMA from that of perfect

scheduling results only from having empty slots during busy periods. Hence, as might be expected (and is shown explicitly in Chapter 2), TDMA performs well for packet generation probability p near 1 and poorly for p near 0. Where applicable, improvement in the TDMA performance can be realized with the addition of a carrier sense assumption, since this reduces the size of an empty slot. A discussion relevant to this situation may be found in [K6].

A very important application of multiaccess protocols is in the computer communications area where the generation of packets by users is characterized as being bursty. Basically, the larger the ratio of a user's average idle period* to average busy period, the more bursty is its packet generation process. Hence it follows that the degree of burstiness is related to the packet generation probability p, the number of users N, and the multiaccess protocol being used. Consider, for example, the TDMA protocol. Through extensions of the results of Section 2.3, we obtain

$$\frac{E[\text{idle period}]}{E[\text{busy period}]} = \frac{1-p}{p}\left[1 + \frac{N}{1-(1-p)^N} - \frac{1}{p}\right]^{-1}$$

Now suppose p = 0.01. When N = 10, the idle to busy ratio is 17.7 and one thus considers the user traffic to be bursty; but when N = 1000, the ratio is 0.110 and, although the packet generation probability p has not changed, the user traffic is no longer considered bursty. Generally speaking, however, for a given protocol and population size N, the user traffic can be made more (less) bursty by making p sufficiently small (large). With

* For the selected user-channel model, we define a user's idle period to begin only when after successfully transmitting a packet in one slot it generates no new packet at the beginning of the next slot.

this in mind, we shall for simplicity, characterize a user as being bursty for p (sufficiently) close to 0 and not so for p (sufficiently) close to 1.

Due to the inadequacy of such conflict-free schemes as TDMA for the case where the packet generation process is bursty, protocols were developed which permit more than one user to transmit in a given slot. These protocols naturally give rise to collisions, but typically perform very well when p is small since, compared to TDMA, the occurrence of empty slots during busy periods can be greatly reduced.

Historically, the Aloha scheme [A1] was the first multiaccess protocol that allowed collisions. This original version, devised and implemented in the context of an unslotted channel, remains today the simplest of the so called random access techniques. With it, a packet is first transmitted at the instant it enters the transmission buffer. When there is a collision, each transmitting user is so informed by not receiving, within some specified time-out period, an acknowledgement packet from the destination user. The waiting time before a collided packet is retransmitted is then selected at random by the user, thus avoiding continually repeated conflicts. The largest attainable throughput for this scheme was determined to be $1/2e \approx 0.18$ [A1]. A slotted version of Aloha followed shortly and resulted in an increase in the maximum throughput to $1/e \approx 0.36$ [R2]. This improvement in performance is a direct consequence of the fact that in a slotted channel colliding packets are forced to overlap completely. Both versions, unfortunately, exhibit unstable behavior. This is manifested through the positive feedback effect of collisions creating more transmissions which in turn may cause still more collisions and so on, eventually driving the system throughput to zero. When this occurs, a restart procedure must be initiated.

The Aloha approach to the multiaccess problem formed the basis for much of the work that followed in protocol development. Part of this effort was directed toward further analysis of the Aloha protocol in terms of its throughput-delay performance [K2,K3] and its characterization as an unstable process [K3,C3]. Other efforts involved improving the Aloha performance through such modeling modifications as carrier sense [K4] and collision detection [M4]. Still other work consisted of devising packet reservation schemes, both implicit and explicit, in which the Aloha concept is used in the reservation process [R3,C5,J1].

Our main concern is with the basic multiaccess problem as characterized by the slotted channel model of Section 1.2. For this situation, Aloha only performs well when p is near 0 and in contrast, TDMA only performs well when p is near 1. What is desired is a protocol whose performance is as close as possible to that of perfect scheduling for all p between 0 and 1. Some efforts along these lines have resulted in protocols which depend on explicit knowledge of the value of p. Other protocols have been developed which, although not directly requiring knowledge of p, either infer this information through channel observations or require more advanced knowledge of the state of the users.

A type of protocol which can be driven by either explicit or inferred knowledge of p is in effect a hybrid of the TDMA and Aloha protocols. The basic concept is to subdivide the population of N users into say L groups of N/L members each. Groups are then assigned slots in a TDMA fashion, and access by the members of a group to its slots is governed by an Aloha protocol. It is the inferred or known value of p which determines the number of groups L. When p is near 0, L = 1 and we have ordinary slotted Aloha. When p is near 1, L = N and we have the usual TDMA protocol. For

intermediate values of p, $1 < L < N$ so that the protocol is a cross between TDMA and Aloha with a performance better than both. Basically, for a given p, L is chosen so as to trade off the empty slots during busy periods generated by TDMA and the collisions generated by Aloha. Specific protocols have been proposed, most formulated for a carrier sense environment, which employ this grouping idea [H2,C4,R1].

Related to this grouping approach is the Tree protocol devised by Capetenakis [C1,C2].* The Tree protocol employs a variable-length frame structure on the time slots and a buffer size of two at each user. A user is allowed to generate at most one packet per frame and all packets generated during one frame are transmitted in the next frame. The size of each frame is determined by the time required to successfully transmit all packets generated during the previous frame. The problem, as usual, is one of not knowing which users generated packets. The approach taken is that of a deterministic tree search based on the ternary channel outcome $\{0,1,\geq 2\}$ observed by all users at the end of each slot. Specifically, at the beginning of each frame the user population is divided into L groups of N/L members each. Suppose for simplicity that N/L is a power of 2. Starting with the first group, all members with packets generated in the previous frame transmit them in the first slot of this new frame. If the slot contains one packet or no packets, then it is determined that at most one user in the group had a packet and, of course, any such packet will have been successfully transmitted. If two or more users have packets, there results a collision. To resolve this conflict the group is divided in half and each half is then treated as a separate group. Hence a collision in either half causes it to be split again, and this continues

* A similar protocol in a polling context was devised by Hayes [H3].

until the slot accessed by a group is empty or contains one packet. This "divide and conquer" technique will resolve all conflicts in the original group of N/L users using at most 2N/L - 1 slots in the new frame. All other groups are handled in an identical fashion and so the frame ends when all packets generated in the previous frame have been successfully transmitted and it is known that no other packets remain.

With the Tree protocol, the group size N/L is chosen at the beginning of each frame to minimize the expected length of the frame. This optimal group size may be expressed directly in terms of the probability that a user generated a packet in the previous frame. This probability is denoted by q and it follows that $q = 1 - (1-p)^{\ell}$ where p is the usual packet generation probability and ℓ is the frame length in slots. When $q \geq 1/\sqrt{2}$, L = N and so the protocol reduces to a form of TDMA [C2]. Also, as expected, L decreases to 1 as q decreases to 0. Note, however, that q is a function of p and if p is not known a priori, it must be estimated from, for example, channel outcome observations.

An important property of the Tree protocol is that, unlike Aloha, it is stable [C1]. This is the result of the increased information contained in the channel feedback. Recall that with slotted Aloha, only a user that transmits in a slot is informed (typically through some acknowledgement mechanism) of whether its transmission was successful. With the Tree protocol, all users are assumed able to determine at the end of each slot whether that slot contains 0, 1, or ≥ 2 packets. Control strategies have been proposed which stabilize slotted Aloha [L1,F1,H1]. These techniques, however, either require additional state information such as knowing the number of busy users (i.e., users with packets) at the beginning of each slot [L1,F1], or require the channel outcome feedback $\{0,1,\geq 2\}$ [H1]. Of

course TDMA requires no feedback information and is certainly stable. However, for an infinite population model, TDMA is not applicable and so an interesting but unsolved problem is that of determining the least amount of feedback necessary for the existence of a stable protocol for infinitely many users.

Underlying the Tree protocol is the concept of a dynamically varying but globally defined "access set". That is, at the beginning of each slot, every user follows a common algorithm, based only on common information, that specifies a subset of users which are given permission to access the slot. Each user in this access set with a packet (generated in the previous frame), then transmits this packet in the slot. With the Tree protocol, the common information is the ternary channel outcomes from previous slots and the common algorithm used is that of a deterministic tree search.

Another protocol employing this access set idea (without the frame structure of the Tree protocol) is the Urn scheme devised by Kleinrock and Yemini [K7]. With the Urn scheme it is assumed that all N users know the number of busy users n at the beginning of every slot. With this information, the size of the access set, k, is computed (in a corrected version of the protocol) according to $k = \lceil (N+1)/n - 1 \rceil$ for $n = 1,\ldots,N$ where $\lceil x \rceil$ denotes the smallest integer greater than or equal to x. The common algorithm at each user which then selects the k members for the access set may take on a number of different forms. The basic algorithm, from which the notion of an urn is derived, is that of a pseudorandom number generator which uses the same seed at each user. As expected, when the traffic is light the access set is quite large (e.g., $n = 1$ implies $k = N$) and as the traffic level increases the size of the access set

decreases. In fact for $n > N/2$, the Urn scheme is equivalent to random TDMA.

The access set selected by the Urn scheme corresponds to an optimal solution to the problem of maximizing the probability of a successful transmission in a slot given that the only information available is the number of busy users (i.e., previous access sets and outcomes are not used in the decision). This is easily proven in two steps. The first step shows that methods for selecting an optimal access set exist within the class of deterministic strategies. A nondeterministic strategy might be one where, for example, each user randomly decides whether it should belong to the access set, as is done with slotted Aloha. Any such randomized strategy, however, may be viewed as a random selection among deterministic strategies. That a deterministic strategy is optimal for this particular problem is a standard result in Bayesian decision theory based on the fact that the maximum value of a random variable is always at least as large as its expected value [D1, Sec. 8.5]. Next, it is necessary to choose from among the 2^N-1 possible access sets. However, due to the homogeneous nature of the user population and the assumption that only the number of busy users is known, it follows that only k, the size of the set, is relevant. Once k is determined, the members may be selected arbitrarily. The optimal k for a given n is readily determined from the element of the hypergeometric density function specifying the probability that only one of the k selected users is busy.

One problem with the Urn scheme is the required global knowledge of the number of busy users at the start of each slot. Such information is clearly not readily available. Kleinrock and Yemini [K7] have proposed that a fixed subchannel be derived (through time-division) from the main

broadcast channel and used in a multiaccess-like fashion in a procedure for estimating n. This represents additional overhead to the protocol in much the same way that a reservation subchannel is overhead to many reservation schemes. Also, although not as catastrophic as with Aloha, the Urn scheme has been shown to be bistable for intermediate traffic levels [M6].

The grouping approaches of the combined TDMA-Aloha, the Tree, and the Urn protocols achieve performance improvements over both basic Aloha and TDMA. Such improvements, however, come at the expense of additional information required in coordinating the user transmissions. In the latter two schemes this leads to questions of robustness which must be effectively dealt with before these protocols can be considered practical.

1.5 Thesis Outline

This thesis is concerned with the multiple access problem as characterized by the user-channel model described in Section 1.2. The results obtained relate either directly or indirectly to the development and analysis of multiaccess protocols for this basic model, but more generally contribute to a better understanding of the multiaccess problem for a variety of user-channel models. In this section we briefly outline the remaining chapters of this thesis.

In Chapter 2, several common measures of steady-state performance for multiaccess protocols are stated and relationships among the measures are derived. From these relationships it is shown that the performance measures are all equivalent in the sense that (1) each performance measure may be expressed as a simple function of any one of the others and (2) a protocol selected to be optimal with respect to any one performance measure is optimal with respect to all of the others. The derived relationships

are also used in the performance analysis of perfect scheduling and TDMA.

Chapter 3 is concerned with the development and, to some extent, the characterization of multiaccess protocols. Here four related classes of multiaccess protocols are defined and examined. First considered is the most general class of protocols, referred to as Team protocols, where the determination of an optimal protocol is formulated as a Team problem. Since general solution techniques to Team problems are nonexistent, we place constraints on the protocol structure that allow a classical sequential decision making formulation of the multiaccess problem. Specifically, we examine three subclasses of Team protocols — the Access Set, Extended Access Set, and Window protocols — where in each case the determination of an optimal protocol can be modeled within the framework of a Markov decision process. However, only with the last class examined, the Window protocols, is the state space finite, and thus amenable to known optimization techniques. The state space for this class is characterized, and properties of the generic protocol and its Markovian structure are derived.

In Chapter 4, Window protocols for the user population sizes N = 2 and N = 3 are constructed. In each case the system state space is first derived along with the associated Markovian decision formulation of the optimization problem. Optimal Window protocols are then found using Howard's policy iteration algorithm [H5]. In addition, the performance of several reasonable but suboptimal Window protocols are computed and compared to that of the optimal.

Chapter 5 considers the problem of designing Window protocols for large user populations. The state space characterizing this situation, however, is enormous (it grows exponentially with N), making standard

optimization techniques impractical. Nevertheless, based on properties of the optimal Window protocols for N = 2 and 3, a reasonable subclass of Window protocols is defined. The finite horizon performance and dynamic behavior of the protocols within this subclass are then investigated. Also, the relationship between the finite and infinite user population problems is examined by considering the limiting behavior of this protocol subclass as $N \rightarrow \infty$.

Finally, Chapter 6 contains conclusions and a discussion of some of the problems that remain to be solved in the multiaccess area.

CHAPTER 2

PERFORMANCE MEASURES AND SOME PRELIMINARY ANALYSIS

Up to now our characterization of the performance of multiple access protocols has been qualitative, based on the extent to which collisions and empty slots during busy periods are avoided. In this chapter, this characterization is made more precise by examining specific quantitative measures of protocol performance. Such measures not only are needed for analytical and numerical comparisons of the effectiveness of different protocols, but are also used in optimizing the performance of a given protocol. In the latter case, the performance measure acts as the objective function in the problem of selecting the best protocol parameter values and/or operational modes for the given system conditions (e.g., given p and N).

2.1 Steady-State Performance Measures

Our main concern is with the long term behavior of multiaccess protocols, and so we restrict our discussion to steady-state performance measures such as throughput and average delay. In this section several common measures of a protocol's steady-state performance are stated and relationships among the measures are derived. From these relationships we show that the measures are all equivalent in the sense that (1) each performance measure may be expressed as a simple function of any one of the others and (2) a protocol selected to be optimal with respect to any one performance measure is optimal with respect to all the others. These results depend only on the user-channel model specified in Section 1.2 (although they are valid independent of any assumed feedback to the users),

and on the existence of limits inherent with steady-state statistics. Since our interest is with protocols for which the given steady-state performance measures exist, this last assumption is by no means restrictive.

Recall that with the single buffer assumption, each user independently generates a packet with probability p at the beginning of each slot, but will only accept a packet into its transmission buffer if the buffer is empty (i.e., if in the previous slot the user either had no buffered packet or had one but successfully transmitted it). A user whose buffer is unable to accept an arriving packet is said to be backlogged and the arriving packet is said to be blocked. Also, each packet in a transmission buffer at the start of a slot is counted as being in the "system" during that slot. With this terminology in mind, consider the following typical steady-state performance measures of a multiaccess protocol:

B_u = E[number of backlogged users]

P_b = Pr[an arriving packet is blocked]

B_a = E[number of blocked packet arrivals per slot]

P_s = Pr[successful packet transmission]

N_s = E[number of packets in the system]

D = E[delay of a packet measured in slots from the time the packet enters a transmission buffer until the end of its successful transmission]

Note that under steady-state conditions, P_s is equal to the system throughput (i.e., the fraction of slots containing successful packet transmissions). Through simple probabilistic arguments we have

$$P_b = B_u/N \tag{2.1}$$

$$B_a = pB_u \tag{2.2}$$

$$P_s = p(N - B_u) \tag{2.3}$$

$$N_s = B_u + P_s \tag{2.4}$$

$$D = N_s/P_s \tag{2.5}$$

where (2.1) and (2.2) follow from the independent but homogeneous nature of packet arrivals, (2.3) follows from the equilibrium condition: E[number of successful packet transmissions per slot] = E[number of unblocked packet arrivals per slot], (2.4) follows after noting that a user with a buffered packet is only backlogged if it is unable to successfully transmit this packet, and finally (2.5) follows from an application of Little's result.

We now show that for any given p and N, by knowing any one of the above six performance measures we can easily determining the others. Using straightforward algebraic manipulations on Equations (2.1)-(2.5), we obtain

$$B_u = \frac{N}{(1 + 1/p(D-1))} \tag{2.6}$$

$$P_b = B_u/N \tag{2.7}$$

$$B_a = pNP_b \tag{2.8}$$

$$P_s = pN - B_a \tag{2.9}$$

$$N_s = N - P_s(1-p)/p \tag{2.10}$$

$$D = \frac{1-p}{p(N/N_s - 1)} \tag{2.11}$$

From these relationships, note that B_u is written as a function of only D and each successive performance measure starting with P_b is written as a function of only the previous performance measure. Hence it follows that after obtaining any one of the given performance measures, the others are easily determined by evaluating simple algebraic equations.

Having shown the relationship among commonly used steady-state performance measures, we now turn to the problem of selecting one for use in comparing multiaccess protocols and/or optimizing the performance of a given protocol. Given p and N, it is clear that desirable protocols would minimize B_u, P_b, B_a, N_s, or D, or maximize P_s. The surprising result that follows from the monotonicity of Equations (2.6)-(2.11) is that a protocol that is optimal with respect to any one of the six performance measures is optimal with respect to the others.

In summary, we have found that each of the given performance measures may be expressed as a simple function of any one of the others, and that the choice of one as a measure for comparing protocols or as an objective function in optimizing the performance of a given protocol is arbitrary. From an analytical point of view these results are significant in that it is often true that a multiaccess protocol is more easily analyzed or optimized with regard to one performance measure than the others. Moreover, we need not be concerned about any trade-off situations where a protocol is optimal with respect to one of the performance measures but not with respect to another.

In the next two sections of this chapter we analyze the performance of perfect scheduling and TDMA. These results will be needed later and are included here because they serve to illustrate the usefulness of results derived in this section.

2.2 Perfect Scheduling Analysis

Perfect scheduling, where both collisions and empty slots during busy periods are eliminated, represents a desired but unattainable level of performance in a system with geographically distributed users and randomly generated packets. As such, its performance provides a useful benchmark for comparing the effectiveness of multiaccess protocols. In this section we analyze the steady-state performance of perfect scheduling for the user-channel model of Section 1.2. The performance measure selected for analysis is N_s, the expected number of packets in the system.

We model the system as a N+1 state Markov chain where the state is equal to the number of users with packets. Under the assumed packet generation process, state transitions occur at the beginning of each slot and are governed by the state transition probabilities

$$p_{0j} = \binom{N}{j} p^j (1-p)^{N-j} \qquad j = 0,\ldots,N$$

$$p_{ij} = \begin{cases} 0 & j = 0,\ldots,i-2 \\ \binom{N-i+1}{j-i+1} p^{j-i+1} (1-p)^{N-j} & j = i-1,\ldots,N \end{cases} \qquad i = 1,\ldots,N \tag{2.12}$$

where p_{ij} is the conditional probability of moving to state j given the system is currently in state i. From these transition probabilities, it follows that the Markov chain is ergodic. Hence the limiting state probabilities π_i, i = 0,1,...,N, defined by

$$\pi_i = \lim_{M\to\infty} \Pr[\text{state} = i \text{ at time-slot } M]$$

exist, are independent of any initial state probability distribution, and are uniquely determined by the equations

$$\pi_j = \sum_{i=0}^{N} \pi_i p_{ij} \qquad j = 0,1,\ldots,N$$

$$1 = \sum_{i=0}^{N} \pi_i$$

Due to the upper Hessenberg structure of the state transition probability matrix $[p_{ij}]$, these equations may be solved in a straightforward manner. A numerically stable algorithm, based on the Q-R decomposition of a matrix [B1,G2], is given in Appendix A.

After solving for the limiting state probabilities, N_s is given by

$$N_s = \sum_{i=0}^{N} i\pi_i$$

Using Equations (2.6)-(2.11), the five remaining performance measures may be computed. In Figure 2-1, all six performance measures are plotted against packet generation probability p for N = 10. Figures 2-2 and 2-3 are graphs of average packet delay D vs. packet generation probability p and average packet delay D vs. throughput P_s, respectively, for various values of the population size N.

Also plotted in Figure 2-3 is the average delay vs. throughput performance of an M/D/1 queue with a one slot service time (see, e.g., [K5]), as characterized by

$$D = 1 + \frac{P_s}{2(1-P_s)} \qquad (2.13)$$

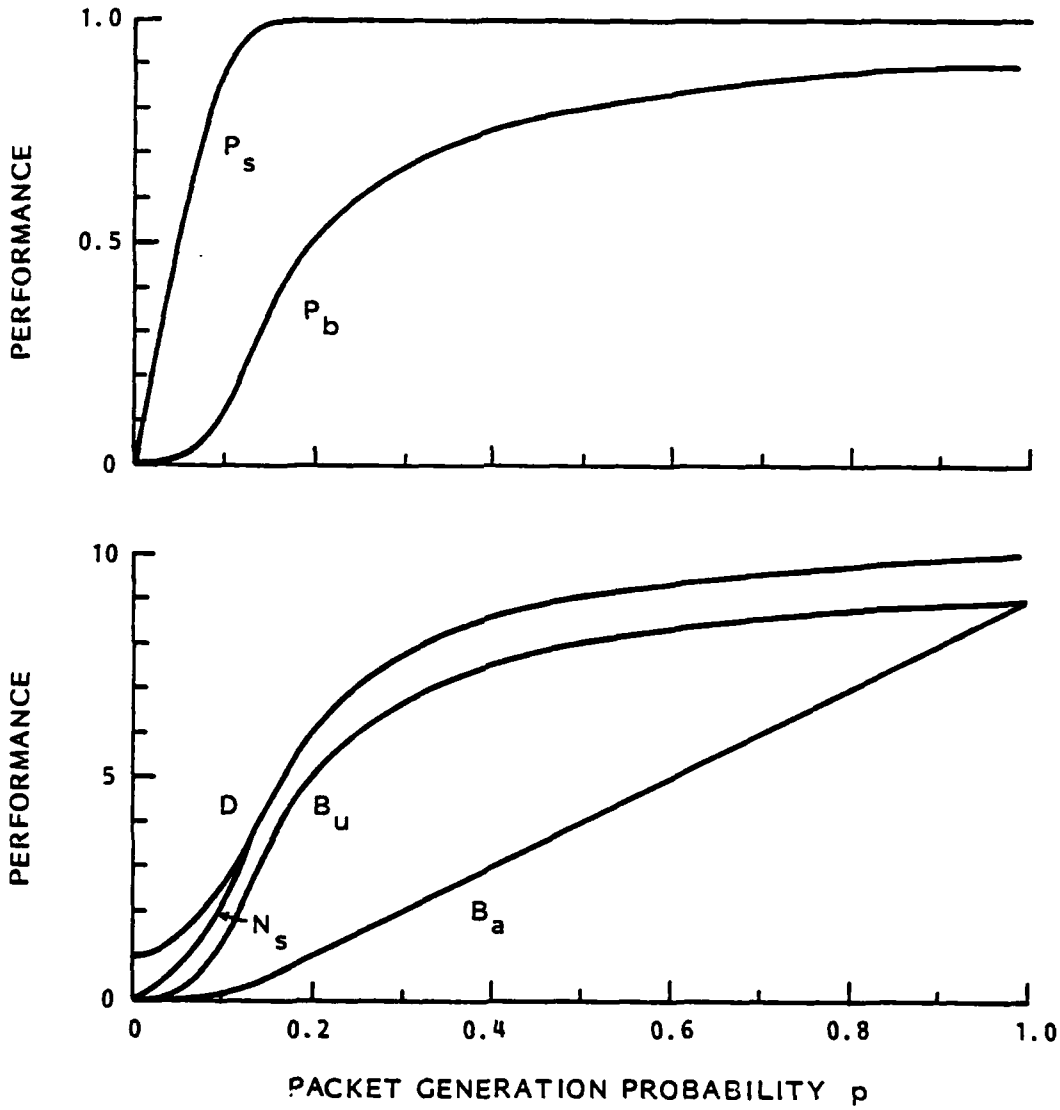


Figure 2-1 Perfect scheduling performance for N = 10

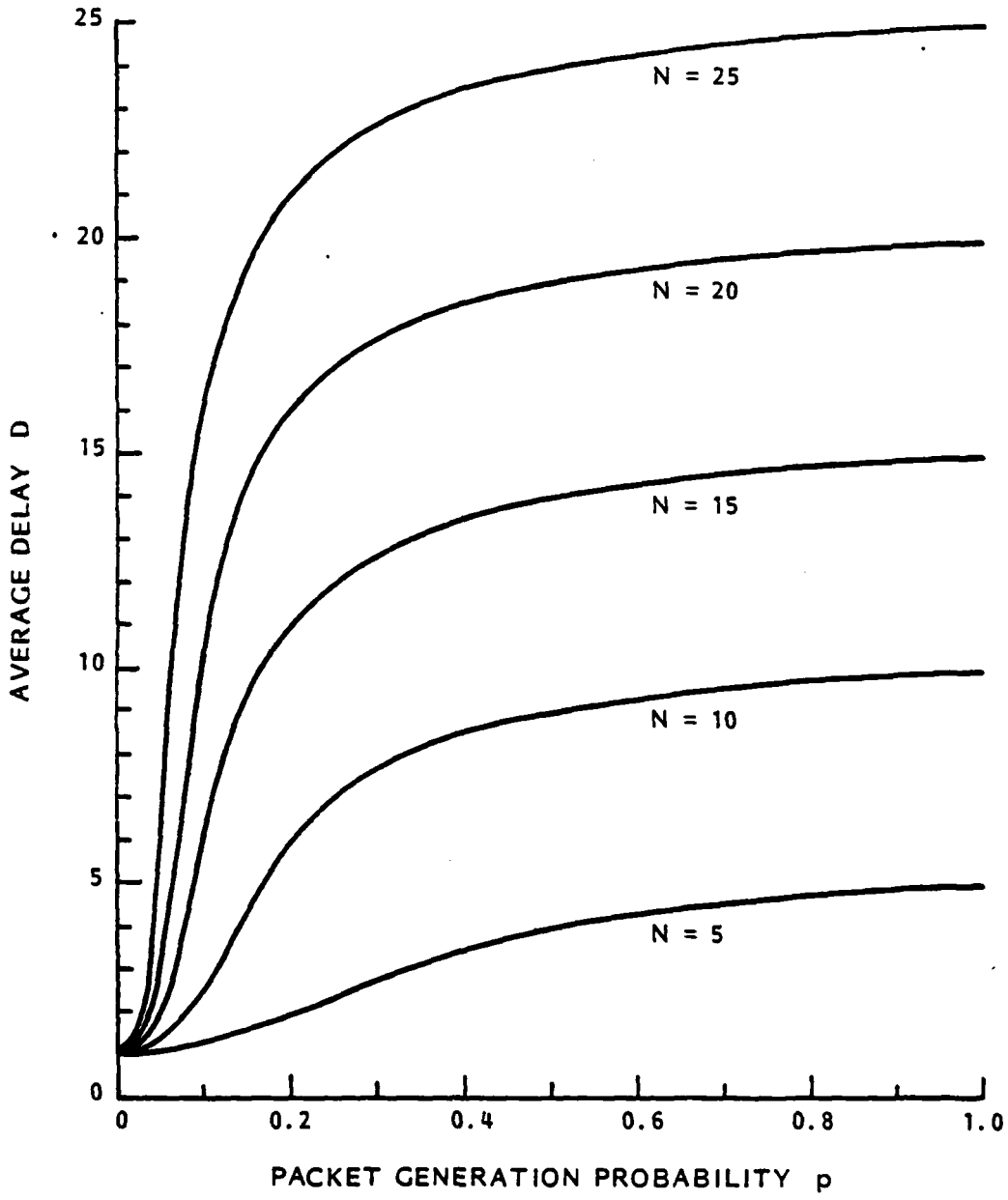


Figure 2-2 Perfect scheduling performance: D vs. p

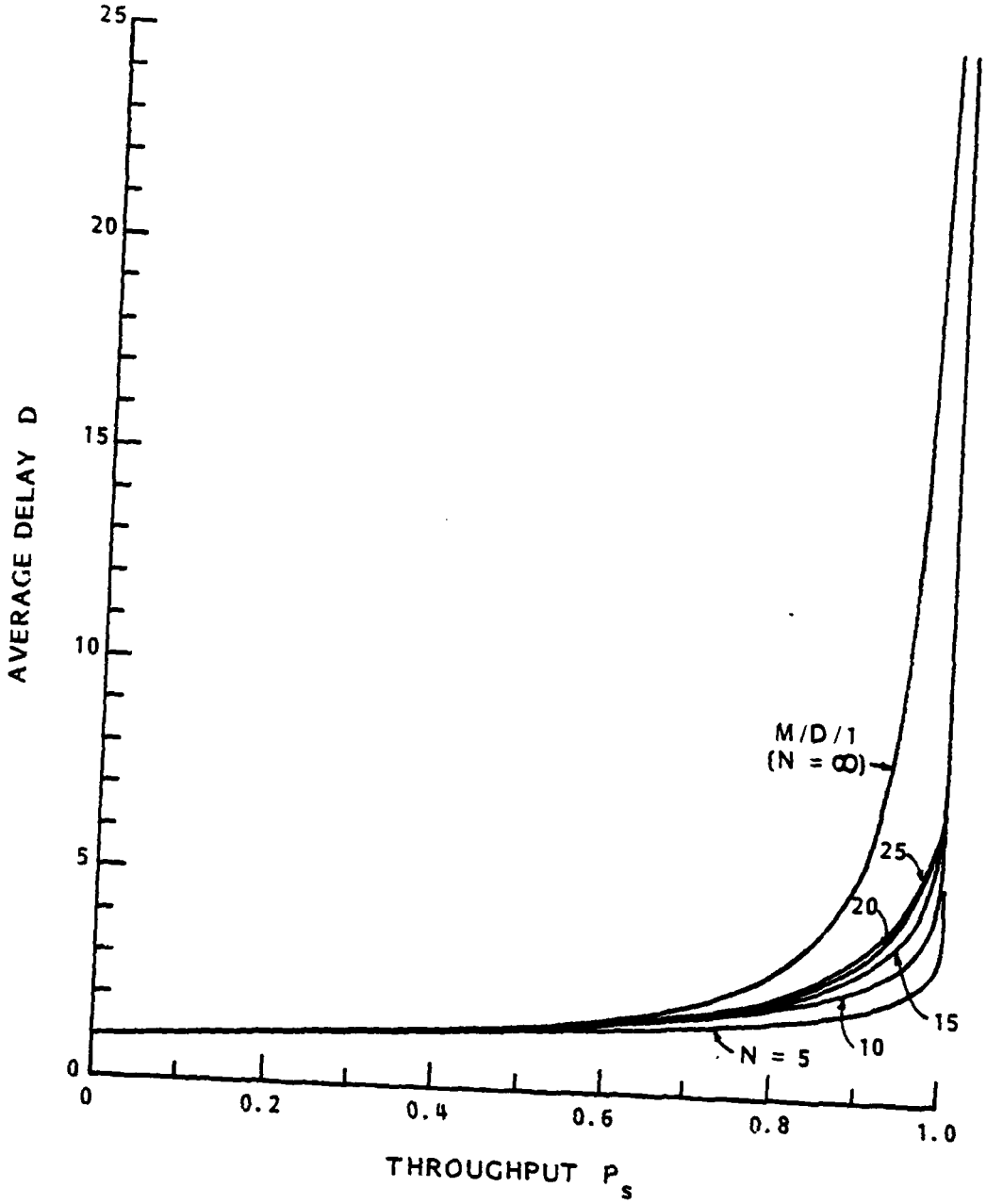


Figure 2-3 Perfect scheduling performance: D vs. P_s

This is shown because it corresponds to the limiting behavior of perfect scheduling for our model as $N \to \infty$ and $pN \in (0,1)$ remains a constant. To see this, first note that in the limit the binomial transition probabilities given in (2.12) become

$$P_{0j} = \frac{\lambda^j e^{-\lambda}}{j!} \qquad j = 0,1,\ldots$$

$$P_{ij} = \begin{cases} 0 & j = 0,\ldots,i-2 \\ \dfrac{\lambda^{j-i+1} e^{-\lambda}}{(j-i+1)!} & j = i-1,i-2,\ldots \end{cases} \qquad i = 1,2,\ldots \tag{2.14}$$

where $\lambda = pN$. That is, the number of new packet arrivals accepted to the system at the beginning of each slot is independent of the current state and has a Poisson density given by

$$\Pr[k \text{ arrivals accepted}] = \frac{\lambda^k e^{-\lambda}}{k!} \qquad k = 0,1,\ldots$$

Also note that in the limit there are no blocked packet arrivals, so that $\lambda \triangleq pN = P_s$. Finally observe that the transition probabilities given by (2.14) correspond to those of the imbedded Markov chain characterizing the state of an M/D/1 queue (with unit service time and arrival rate λ) at the departure times of the system. Hence paralleling an analysis of the M/D/1 queue, we obtain (2.13).

2.3 TDMA Analysis

The performance analysis of TDMA for the user-channel model of Section 1.2 is easily obtained. Recall that with TDMA, each user is given exclusive transmission rights to a slot once every N slots. It follows that P_s, the system throughput, is equal to the probability that a user has generated a packet by the start of its assigned slot. Hence we have

$$P_s = 1 - (1-p)^N \tag{2.15}$$

Using Equations (2.6)-(2.11), expressions in terms of only p and N may be derived for the five remaining performance measures. For example, from (2.10) and (2.11) we obtain

$$D = 1 + N/P_s - 1/p$$

and thus from (2.15), the average delay for TDMA is given by

$$D = 1 + \frac{N}{1-(1-p)^N} - \frac{1}{p} \tag{2.16}$$

Although one may derive (2.16) directly, it is not as trivial as determining the throughput P_s and then applying the results of Section 2.1.

In Figure 2-4, the six performance measures are plotted against p for N = 10. Figures 2-5 and 2-6 are graphs of D vs. p and D vs. P_s, respectively, for various values of N.

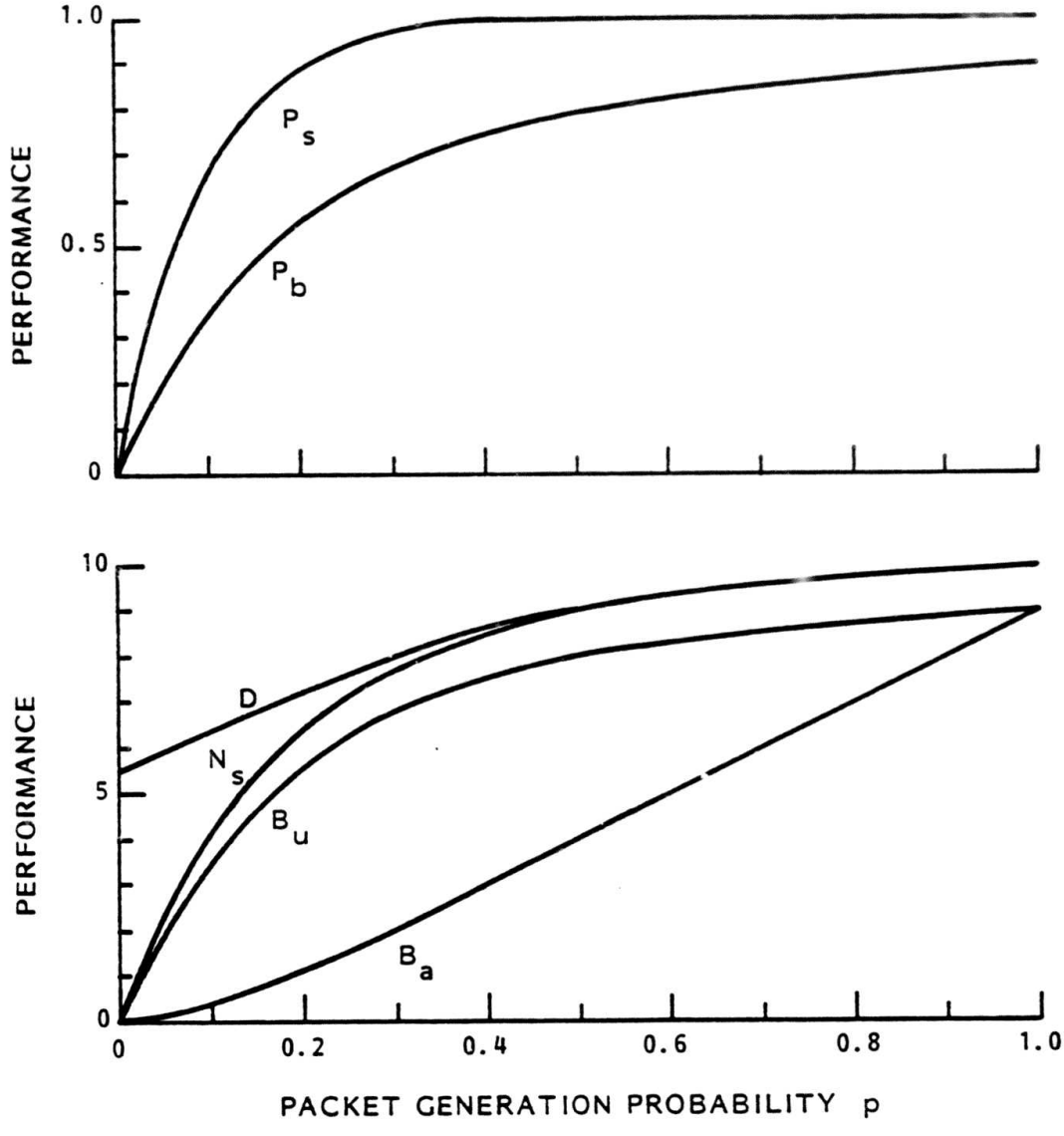


Figure 2-4 TDMA performance for N = 10

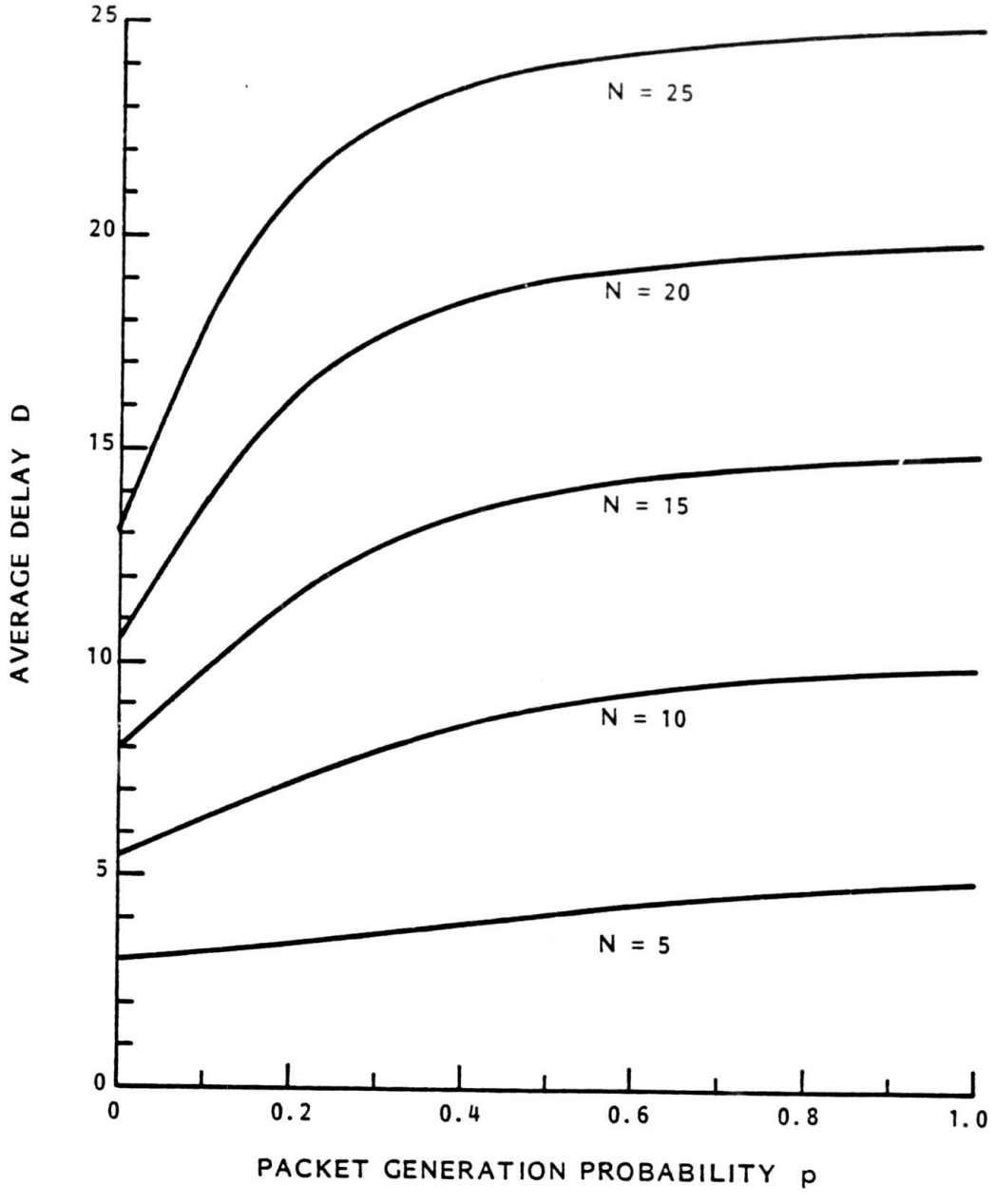


Figure 2-5 TDMA performance: D vs. p

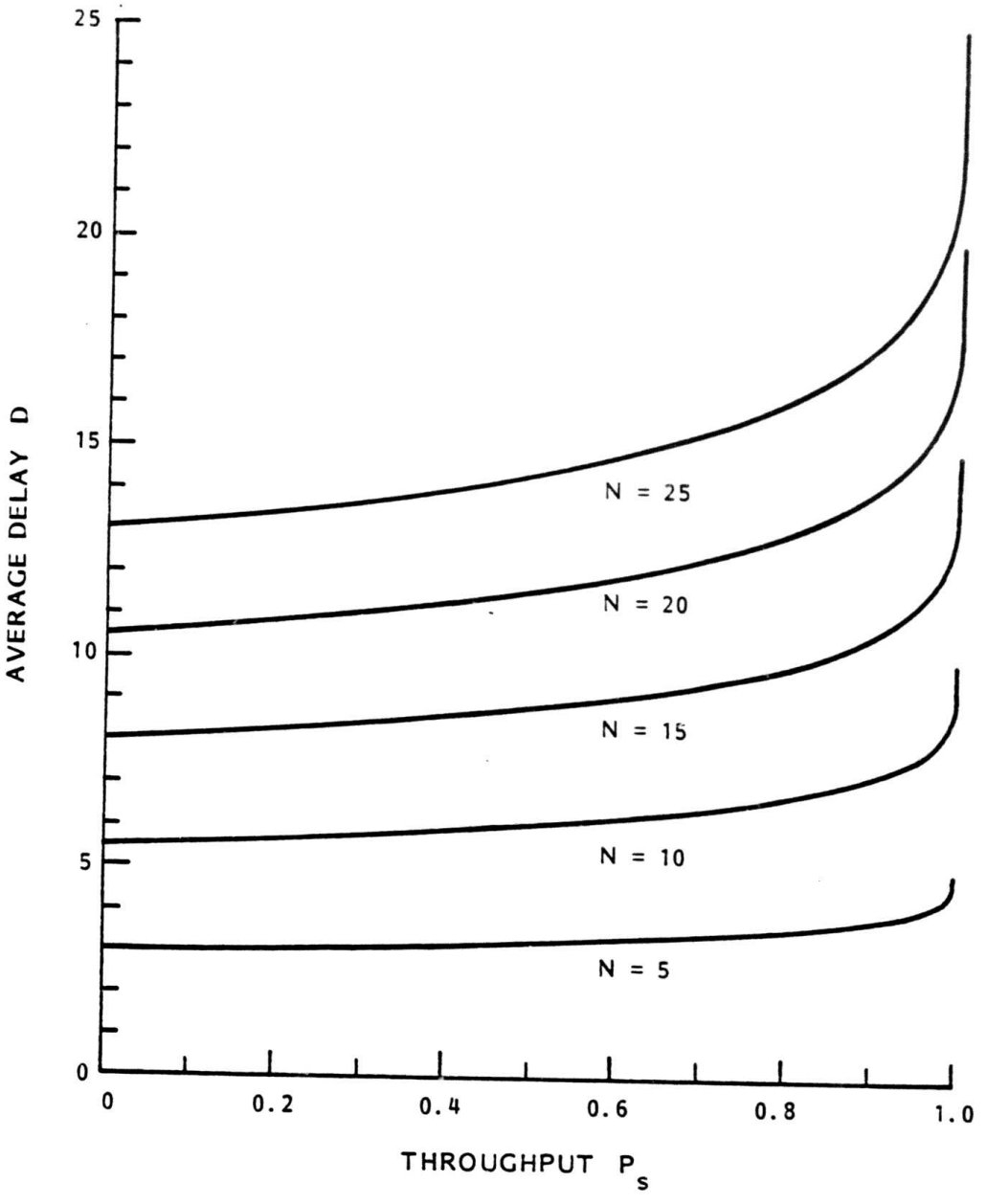


Figure 2-6 TDMA performance: D vs. P_s

CHAPTER 3

MULTIACCESS PROTOCOL DEVELOPMENT

This chapter is concerned with the development of multiaccess protocols for the user-channel model of Section 1.2. Illustrated in Figure 3-1 are the various classes of protocols examined. The Team protocols constitute the most general class and are so named because to determine the most efficient protocol within this class is a Team theoretic problem. The Access Set and Extended Access Set protocols are two restricted but reasonable subclasses of Team protocols allowing a classical sequential decision making formulation of the multiaccess problem. Finally, the class of Window protocols is a subclass of the Extended Access Set protocols whose state space is finite; and thus, as we shall see, one to which known optimization techniques can be applied. We begin our discussion with the class of Team protocols.

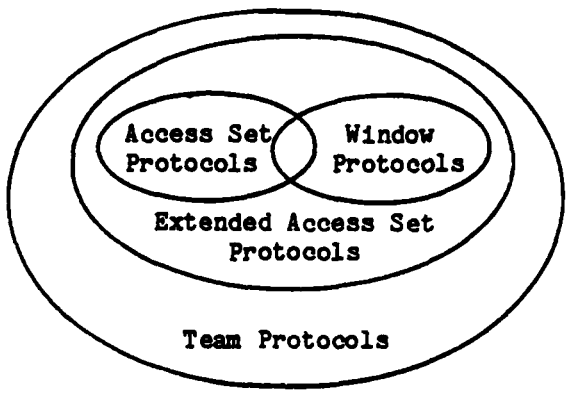


Figure 3-1 Classes of multiaccess protocols

3.1 Team Protocols

Consider the underlying structure of the generic multiaccess protocol for the user-channel model we have specified. At the beginning of each slot, based on its current knowledge of the state of the system, each user with a buffered packet must decide whether to transmit its packet in the slot. A user's knowledge of the "state of the system" may, in general, be based on all the information that is available to it, including the feedback obtained from previous channel outcomes (common information) and the past history of its own packet arrivals and transmission decisions (local information). Moreover, considering the performance measures we have selected, a user's decision to transmit or not is made unselfishly, with the goal being to optimize some global objective function. Such a problem of sequential decision making in an environment of decentralized decision makers with distributed information and a common objective function may be formulated within the framework of Team theory [M2,H4].

The notion of a dynamic Team problem has been around for over 25 years [M1]. Unfortunately, the class of problems is of sufficient complexity that little progress has been made toward a general solution technique or even in finding general properties of optimal solutions. Hence its value to the multiaccess problem, even with the relatively simple user-channel model of Section 1.2, does not go much beyond a conceptual level.

Without established solution methodologies, one is forced to restrict the scope of feasible solutions to those classes to which known optimization techniques can be applied. In the next three sections of this chapter we examine three related subclasses of Team protocols: the Access Set, Extended Access Set, and Window protocols. Each class can be modeled as a Markov decision process [D2,H5,R4], but only with the latter can we

generally solve for the optimal protocol.

3.2 Access Set Protocols

The concept of an access set was introduced in describing the Tree [C2] and Urn [K7] protocols in Section 1.4. The basic idea is that, at the beginning of each slot, every user follows a common algorithm, based only on common information, that specifies a subset of users which are given permission to access the slot. Each user in this access set with a packet then transmits its packet in the slot. The sequential nature of the process is illustrated in Figure 3-2 where A(j) is the access set for slot j, T(j) is the subset of users in A(j) which transmit packets in slot j, and C(j) is the common channel observation which for our model corresponds to the ternary channel outcome $\{0,1,\geq 2\}$ observed at the end of slot j.

The above structure imposes a form of coordination among the users in which both common and local information are employed in a user's decision to transmit a packet. The channel outcomes are common information and are used in selecting the access set. The local information consists of each

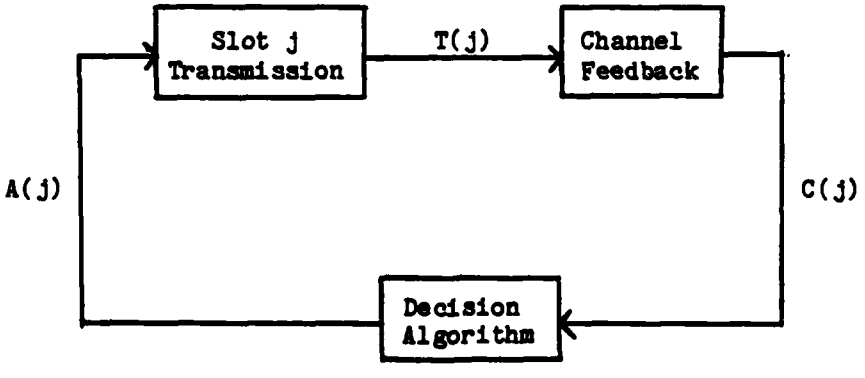


Figure 3-2 Structure of Access Set protocols

user knowing whether it has a packet and thus whether to transmit given that it is in the current access set. The use of the local information is predetermined since by definition a user in the access set is required to transmit if it has a packet. What remains to be specified is the decision algorithm used to determine the access set A(j) at the beginning of each slot j. Since both the algorithm and its inputs are restricted to be common to all users, the problem may be formulated in the context of classical (i.e., nondistributed) sequential decision making [B2].

With the classical sequential decision making formulation, the information available to the decision process for selecting A(j) are the previous observations C(1),...,C(j-1) and decisions A(1),...,A(j-1) along with the given initial conditions of the system. The decision A(j) may, in general, be a probabilistic function of this past history of the system. However, we require all users to compute the same access set A(j) for each j, and hence any randomization in the decision by the algorithm must have the same outcome at each user. This may be accomplished with the use of identical, precomputed tables of samples from appropriate probability distributions stored at each user; or, for a more practical method, one might consider using a pseudorandom number generator with the same seed at each user. Such a "centralized" structure for randomizing decisions is in reality more general than allowing users to independently randomize their own decisions. To see this, note that one type of centralized structure consists of choosing an independent random decision for each user; in effect each user has knowledge of the other decisions (and thus of the access set) but does not use this knowledge. Hence, in searching for an efficient Access Set protocol, we may restrict our attention to the class of centrally randomized decisions which includes as a subclass all

deterministic decision algorithms. Later when we examine the class of Window protocols, we shall see that there exists an optimal Window protocol whose decision process is deterministic.

The steady-state performance measures examined in Chapter 2 correspond to the infinite horizon average expected value problem in the sequential decision making nomenclature. Due to their equivalence, any one of the six may be chosen as the reward (cost) function for our problem. One that is easily incorporated into the problem formulation we develop is P_s, the system throughput. Defining the immediate reward

$$r(j) = \begin{cases} 1 & \text{if slot } j \text{ contains a successful transmission} \\ 0 & \text{otherwise} \end{cases}$$

we have, assuming the limit and expectation exist,

$$P_s = \lim_{M \to \infty} \frac{1}{M} \; E\left[\sum_{j=1}^{M} r(j) \right] \tag{3.1}$$

where the expectation is conditioned on both the selected decision algorithm and the given initial conditions of the system. Adopting notation from sequential decision making, we shall occasionally refer to the decision algorithm as a <u>policy</u> and the decision A(j) as a <u>control</u>. The problem of interest is that of determining, for any given p and N, a policy which maximizes (3.1).

To develop a framework for finding an optimal policy, we begin by defining the internal state vector $u(j) = (u_1(j),\ldots,u_N(j))$ where

$$u_i(j) = \begin{cases} 1 & \text{if user } i \text{ has a packet at the beginning of slot } j \\ 0 & \text{otherwise} \end{cases}$$

Note that u(j) simply indicates which users have packets ready for transmission at the beginning of slot j. For the packet generation process specified in Section 1.2, internal state transitions can be modeled by a 2^N-state discrete-time Markov chain where the probabilities governing the transition to u(j+1) depend only on u(j) and the control A(j). To better understand this, consider the event sequence depicted in Figure 3-3. The first stage corresponds to the end of slot j and the last corresponds to the beginning of slot j+1. The transition from u(j) to u(j+1) occurs between the first and last stages and may be divided into two steps. First, depending on the access set A(j), at most one user will successfully transmit a packet during slot j so that for at most one i, $u_i(j)$ goes from 1 to 0. Second, each user which does not currently have a packet (including user i) will generate one with probability p. Hence we see that transition to state u(j+1) is a probabilistic function of only the current

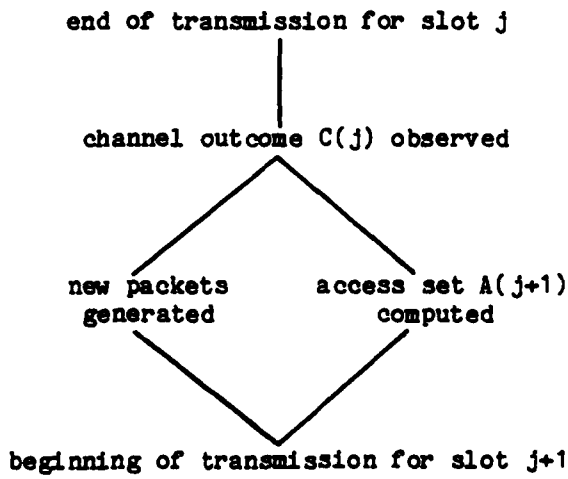


Figure 3-3 Sequence of events from the end of slot j to the beginning of slot j+1

state u(j) and access set A(j); and thus u(j) is a controlled Markov process.

Note that the internal state u(j) is not available to the decision process at the beginning of each slot j. Clearly if it were, then the decision algorithm could be selected to achieve perfect scheduling of all packet transmissions. The problem of optimally controlling the Markov process u(j) may, however, be formulated in terms of a partially observable Markov decision process [A2,S1]. This follows since after the decision algorithm selects an access set A(j), an output $C(j) \in \{0,1,\geq 2\}$ is observed, a reward r(j) is earned, and a transition is made to a new internal state u(j+1); where the observed output, reward, and transition depend only on the current internal state u(j) and decision A(j). As previously indicated, in addition to the given initial conditions of the system, the information available to the decision process for selecting A(j) are the previous channel outcomes C(1),...,C(j-1) and access sets A(1),...,A(j-1). It is a standard result that the 2^N-vector $\pi(j)$, where component $\pi_i(j)$ is the conditional probability of being in internal state i at the beginning of slot j given the above previous inputs, outputs, and initial conditions of the system, is a sufficient statistic for the complete past history of the process. Moreover, from Bayes' Rule it follows that $\pi(j)$ may be expressed as a function of only $\pi(j-1)$, A(j-1), and C(j-1) and thus computed recursively. Hence, $\pi(j)$ can be viewed as the state of a discrete-time Markov decision process upon which the decision A(j) is based.

The difficulty we now face in determining an optimal policy (i.e., a function mapping $\pi(j)$ into A(j) which maximizes (3.1)) stems from both the type of performance measure we have selected and the new state space for

the process. It is well known that if the state and control spaces are finite, then an optimal policy for an infinite horizon average expected value problem exists and is in the class of stationary deterministic policies (i.e., the mapping is nonrandom and independent of j).* Moreover, techniques such as Howard's policy iteration algorithm [H5] exist for determining such an optimal policy. However, if the state space is allowed to be infinite, then optimal policies may not exist or, when they do exist, they may not be stationary or deterministic [B2,D2,R4]. Now although the internal state u(j) is from a finite state space (having 2^N elements), the state space, H, corresponding to the new problem is generally infinite. To see this, consider the case where the decision algorithm always selects user 1 (i.e., A(j) = {1} for all j). For this policy it follows that at the beginning of slot j, each user i independently has a packet with probability P_i where

$$P_i = \begin{cases} p & i = 1 \\ 1 - (1-p)^j & i = 2,\ldots,N \end{cases}$$

and hence H is at least countably infinite.

3.3 Extended Access Set Protocols

It is of value to note that the class of Access Set protocols may be extended while maintaining the classical sequential decision making formulation of the multiaccess problem. Specifically, one might consider controlling packet transmissions via a time interval mechanism in addition

---

* This is assuming that we have control over the starting state of the system. Without this assumption we would require an additional condition such as that every stationary policy results in an indecomposable Markov chain (i.e., the chain contains exactly one irreducible set of states, the rest being transient).

to the access set. That is, a user's packet is transmitted only if the user is in the access set and the packet was generated in some globally defined time interval (or intervals), where both the access set and time interval(s) are computed by each user according to some common algorithm based only on common information.*

To see how such an extension to the Access Set protocol might be useful, consider the case of three users where $u(j) = (0,1,1)$ and $A(j) = \{1,2,3\}$ so that the channel outcome $C(j)$ indicates a collision. Now suppose that user 1 generates a packet at the beginning of slot $j+1$, so that $u(j+1) = (1,1,1)$, and that the decision algorithm sets $A(j+1) = \{1,2\}$. There of course will be another collision and user 1 knows this a priori since from the previous slot it was able to determine that both users 2 and 3 have packets. Hence, even though user 1 is in the access set, by not transmitting it will prevent a collision and ensure a successful transmission. This problem may be avoided with the addition of the time interval mechanism. Specifically, by having the decision algorithm also select $[1,j]$ as the time interval for slot $j+1$, the packet generated at the beginning of slot $j+1$ by user 1 would not be allowed to be transmitted in slot $j+1$ and hence no collision would ensue.

The additional control provided by the time interval mechanism allows further flexibility in the design of a multiaccess protocol over that of the basic Access Set structure, without precluding a Markovian decision formulation of the problem. One may, for example, take the internal state to consist of the set of users with packets and how long each such user has had its packet awaiting transmission. Then, as we did with the Access Set protocols, the decision making may be formulated in terms of a partially

---

* A variation on this extension would replace "and" with "or".

observable Markov decision process. Such an extension does, of course, further complicate the already difficult problem of finding an optimal protocol. In the next section, however, we examine a subclass of these Extended Access Set protocols where the Markov decision formulation has a finite state space, and thus one for which an optimal policy can be determined.

Finally, it is worth noting that, aside from variations in the assumed feedback, many of the currently proposed multiaccess protocols may be viewed as being from the general class of Access Set or Extended Access Set protocols. Two simple examples are TDMA [M3] and slotted Aloha [K3]. With both protocols there is no assumed feedback of common information to the users and so the decision process runs open loop. The decision process is deterministic for TDMA: access sets contain one member and users are assigned to access sets in a round-robin fashion. For slotted Aloha, the decision process is random in the distributed sense: each user independently decides by "flipping a biased coin" whether to belong to the access set.* The Urn protocol [K7] and especially the Tree protocol [C2] are more in line with the type of protocol we have been discussing, since with both, the access set is selected based on the feedback of common information to all users. With the Urn protocol the access set is selected in a centrally randomized fashion. With the Tree protocol the decision process is deterministic, and since, with its frame structure, packets generated during one frame cannot be transmitted until the next, the protocol is a member of the Extended Access Set protocols. Lastly,

---

* If the protocol permits those users which have generated packets at the beginning of the slot to transmit with probability one, then we may view the protocol as being a member of the "or" version of the Extended Access Set protocols.

although designed for an infinite population model, Gallager's multiaccess protocol [G1] selects users to transmit by a time interval mechanism alone, and hence may be considered a member of the Extended Access Set protocols.

3.4 Window Protocols

The class of protocols discussed in this section use a windowing operation for selecting the access set. Specifically, the N users are ordered (algorithmically speaking) on a circle as illustrated in Figure 3-4 and the access set is selected by a window that rotates around the circle. That is, at the beginning of each slot, the access set for that slot consists of all users within the window (e.g., in Figure 3-4, $A(j) = \{3,\ldots,6\}$). As for the movement of the window, if a collision occurs, the tail of the window remains fixed and the window size decreases. After an empty slot or a success, the tail of the window advances along the circle

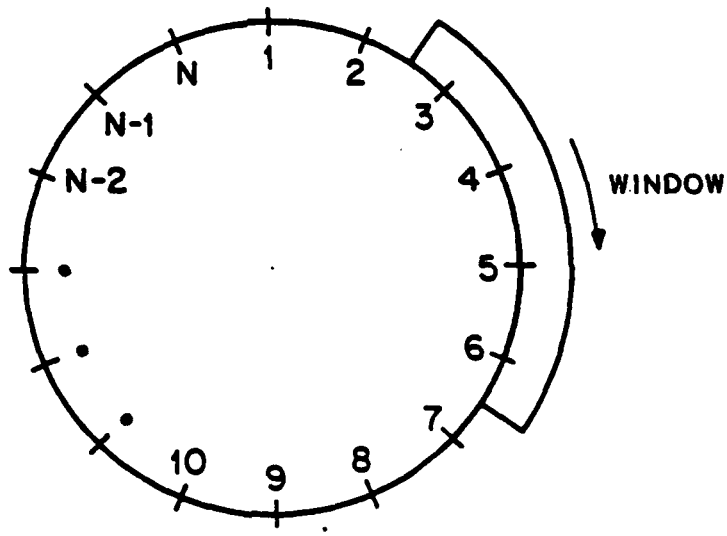


Figure 3-4 Selection of an access set via a windowing operation

to the head of the previous window with the window size possibly changing. Note that the protocol is inherently fair in that for each revolution of the window every user is given the opportunity to successfully transmit one packet. Also, the window approach to selecting the access set simplifies the decision algorithm, since the only decision to be made at the beginning of each slot is the window size. As an indication of its intuitive appeal, this basic windowing concept was independently proposed as an extension to the Tree protocol by Gallager [G1] and the Urn protocol by Kleinrock and Yemini [K7].

3.4.1 Protocol Description

The class of Window protocols defined in this section have additional restrictions on how the window size changes and, using a time interval mechanism, on which packets generated by users in the window are allowed to be transmitted. These restrictions actually only occur after a collision, whereupon the operation of the window protocol enters a conflict resolution mode. It is instructive to consider first the situation where there are no collisions and then afterward the general case.

Suppose each access set selected by the window results in either a successful transmission or an empty slot. It follows then that each user i will independently have a packet with probability

$$P_i = 1 - (1-p)^{T_i} \qquad (3.2)$$

where T_i is the positive integer number of slots since user i was last included in the window. If we continually renumber the users so that user 1 is always the first user in the window and user 2 is the next clockwise to 1 and so on, then clearly

$$T_1 \geq T_2 \geq \cdots \geq T_N \tag{3.3}$$

so that

$$P_1 \geq P_2 \geq \cdots \geq P_N$$

Hence each user in the window has at least as high a probability of possessing a packet as any user not included in the window. As evidence of the reasonableness of selecting the access set through a window protocol, it is shown in Appendix B that the subset of the N users which maximizes the probability of a successful transmission is of the form $\{1,2,\ldots,k\}$ for some $1 \leq k \leq N$ (assuming that each user independently has a packet with probability P_i and $P_1 \geq P_2 \geq \cdots \geq P_N$).

When there is a collision, the protocol enters a conflict resolution mode (steps 2 and 3 in the description). During this phase a restricted class of users R is specified before the start of each slot. The restriction is that any packet a user generates while in R cannot be considered for transmission until after the user leaves R. This constraint on the protocol is made to maintain a tractable state space, but is also intuitively reasonable since allowing new packets to enter the conflict resolution process can only increase the uncertainty as to which users were originally involved in the collision.

The generic operation of the Window protocol is given in algorithmic form in Figure 3-5. For notational convenience we number the users from 0 to N-1 (there is no renumbering in this description as the window changes), and we define the subset of users

```
step 1.  W = [i,j], R = ∅
           if empty or success
             d.a. selects m ∈ {1,2,...,N}
             i ← j+1
             j ← j+m
             go to step 1

           if collision
             d.a. selects k ∈ {i,i+1,...,j-1}
             go to step 2

step 2.  W = [i,k], R = [i,j]
           if empty
             i ← k+1
             d.a. selects k ∈ {i,i+1,...,j-1}
             go to step 2

           if success
             i ← k+1
             d.a. selects k ∈ {i,i+1,...,j}
             go to step 3

           if collision
             j ← k
             d.a. selects k ∈ {i,i+1,...,j-1}
             go to step 2

step 3.  W = [i,k], R = [i,j]
           if empty
             i ← k+1
             d.a. selects k ∈ {i,i+1,...,j}
             go to step 3

           if success
             d.a. selects m ∈ {1,2,...,N}
             i ← k+1
             j ← k+m
             go to step 1

           if collision
             j ← k
             d.a. selects k ∈ {i,i+1,...,j-1}
             go to step 2
```

Figure 3-5 Window protocol operation

$$[i,j] = \begin{cases} i & i = j \\ i,i+1,\ldots,j & i < j \\ i,i+1,\ldots,N-1,0,1,\ldots,j & i > j \end{cases}$$

The first line of each of the three steps denotes the control as specified by the window W and restricted class R (Ø denotes the empty set). Following this is the action taken by the decision algorithm (d.a.) for each of the possible channel outcomes {empty, success, collision}. The process starts at step 1 with no outstanding collisions to resolve, and all additions (+) are computed modulo N. In reading through the algorithm, it is helpful to keep in mind that at step 2 and step 3 there are, respectively, ≥ 2 and ≥ 1 users in R with packets. A sequence of feasible window W and restricted class R changes is illustrated in Figure 3-6; where, for convenience, the circle is cut between users 0 and N-1 and extended along a straight line, and packet arrivals and departures to a user's buffer are denoted by arrows above and below the line, respectively.

Note from Figure 3-5 that an empty slot or a success always causes the tail of the window to advance to the head of the previous window, and a collision always causes the tail to remain fixed and the window size to decrease. There are no restrictions on how much the window size reduces following a collision, and the only real restriction on the selected window size following an empty slot or a success occurs when entering step 3. Here the protocol requires that $W \subseteq R$. We conjecture, however, that this restriction results in no degradation in performance for this class of protocols.

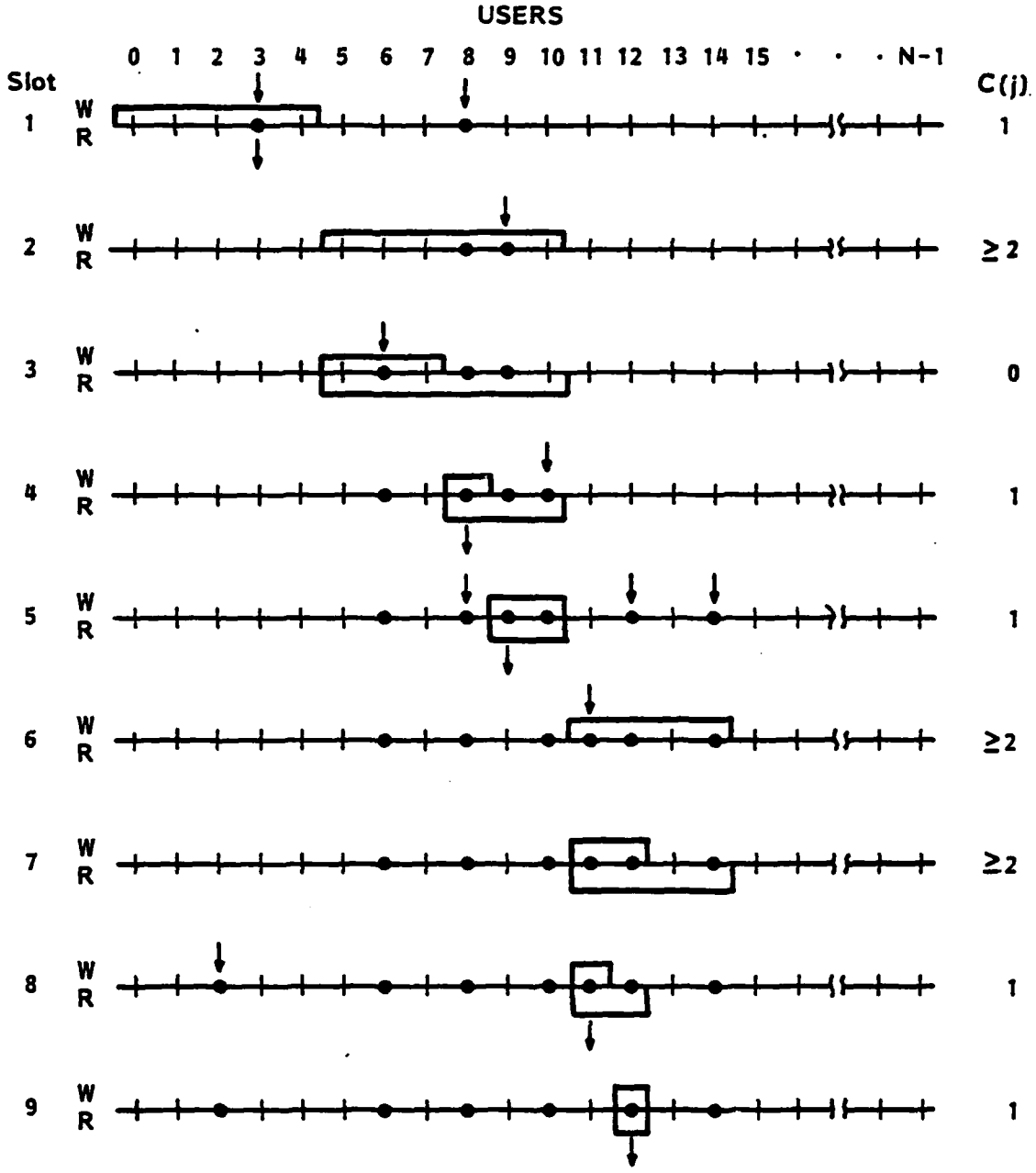


Figure 3-6 Example of Window protocol operation

3.4.2 Protocol State Space

The positive integer variable T_i, introduced in (3.2) for the case of no collisions, is a convenient mechanism for tracking the system state upon which the window size decisions are based. It is updated for each user i at the end of every slot following the observation of the channel outcome. The update rules are given in Figure 3-7 where

$$\tau = 1 + \text{number of slots since R last became nonempty} \qquad (3.4)$$

Now from the protocol description it follows that each user $i \notin R$ independently has a packet with probability P_i given by (3.2) where T_i is determined from the update rules given in Figure 3-7. This is true even though, after a success or collision at step 3 when $k \neq j$ or a collision at step 2, the users in the set $[k+1,j] = R-W$ are removed from R before being processed* by the window. To see this, consider the case of a collision at step 1 followed by another collision at step 2. Letting $[i,j] \geq m$ denote the event that the user set $[i,j]$ has $\geq m$ users with packets, we have

$$\begin{aligned} &\Pr\{[k+1,j] \geq m \mid [i,j] \geq 2,\ [i,k] \geq 2\} \\ &\quad = \Pr\{[k+1,j] \geq m \mid [i,k] \geq 2\} \qquad (3.5) \\ &\quad = \Pr\{[k+1,j] \geq m\} \qquad (3.6) \end{aligned}$$

where (3.5) follows since $[i,k] \geq 2$ implies that $[i,j] \geq 2$, and (3.6) follows from the independence of the users in the disjoint sets $[i,k]$ and $[k+1,j]$. Hence the statistics associated with users $[k+1,j]$ after $W = [i,k]$ (but before the T_i's are updated) are identical to what they were immediately before the users entered R. A similar argument can be made for

---

* A user is "processed" when the tail of the window advances past the user on the circle.

(1) $i \notin W$, $i \notin R$
$T_i \leftarrow T_i + 1$

(2) $i \in W$, $i \notin R$
if empty or success
$T_i \leftarrow 1$

if collision
no change

(3) $i \notin W$, $i \in R$
if success or collision at step 3 or collision at step 2
$T_i \leftarrow T_i + \tau$

otherwise
no change

(4) $i \in W$, $i \in R$
if empty or success where user i did not transmit
$T_i \leftarrow \tau$

if empty or success where user i transmitted
$T_i \leftarrow 1$

if collision
no change

Figure 3-7 T_i update rules for Window protocol

the case of a success or collision at step 3 when $k \neq j$. This feature of the protocol is also present in an analogous form in the multiaccess protocol devised by Gallager [G1], and a simpler version exists in the group testing algorithm of Sobel and Groll [S2] which has recently been applied to the multiaccess problem by Towsley and Wolf [T2].

Now, although the independence property associated with each user $i \notin R$ does not hold for users that are currently in R, their contribution to the system state is easily characterized by T_i for each $i \in R$, the

current value of τ (generally needed to update T_i when user i leaves R), and whether there are ≥ 2 (when at step 2) or ≥ 1 (when at step 3) users in R with packets. Hence when R is empty (i.e., at step 1 of the protocol) the system is completely characterized by the vector $(T_1,T_2,\ldots,T_N)$; and when R is not empty, the characterization requires, in addition to $(T_1,T_2,\ldots,T_N)$, the set R, the value of τ as defined by (3.4), and whether there are ≥ 1 or ≥ 2 users in R with packets.

There are two aspects of the Window protocol to be discussed before continuing with the state space discussion. First, note in case (4) of the T_i update rules that to compute the new value for T_i following a successful transmission requires the identity of the user that transmitted the packet. This is typically not a problem for a real communication system, but nevertheless represents additional input to the decision process in order for it to keep track of the system state. Second, also from case (4) note that to maintain the ordering of the T_i's as specified in (3.3) (assuming the renumbering of users which gives rise to (3.3)), and thus in a sense the fairness of the protocol, requires that users be occasionally reordered on the circle. For example, consider the case of three users where at step 1, u(j) = (1,0,1) and the following sequence of window decisions and channel outcomes occur:

$$W = \{1,2,3\} \xrightarrow{C} W = \{1\} \xrightarrow{S} W = \{2,3\} \xrightarrow{S}$$

Note that after the second success the protocol will be back at step 1 and $(T_1,T_2,T_3) = (2,3,1)$. Thus, to be fair in the sense of (3.3), users 1 and 2 should be interchanged on the circle before the start of the next slot. Both of the above complications result from the packet transmission restrictions that stem from R.

Let G_1, G_2, and G_3 denote the three classes of states corresponding to, respectively, steps 1, 2, and 3 in the Window protocol description. To simplify the state space, we dynamically renumber the users so that user 1 always corresponds to the first user in W and user 2 is the next clockwise to 1 and so on. Also, to maintain the fairness of the protocol, when necessary we reorder the users on the circle upon entering step 1 so that

$$T_1 \geq T_2 \geq \cdots \geq T_N$$

Hence, corresponding to the three steps in the Window protocol description, we have the following three types of states:

G_1 state: $(T_1, T_2, \ldots, T_N)$

G_2 state: $(T_1, T_2, \ldots, T_N; 2, \ell, \tau)$, $\ell \in \{2,3,\ldots,N\}$

G_3 state: $(T_1, T_2, \ldots, T_N; 3, \ell, \tau)$, $\ell \in \{1,2,\ldots,N-1\}$

where $R = \{1,2,\ldots,\ell\}$ and τ is measured relative to the end of the slot during which the system is in the defined state so that $\tau \geq 2$. Note that when R contains exactly two users at step 2, we have that each user $i \in \{1,2\}$ has a packet awaiting transmission with probability one, independent of the specific values of T_1 and T_2. Likewise when R contains exactly one user at step 3, user 1 has a packet with probability one independent of T_1. Hence, to further simplify the state space, we denote the state for these specific step 2 and step 3 cases by $(\infty, \infty, T_3, \ldots, T_N)$ and $(\infty, T_2, T_3, \ldots, T_N)$, respectively.

Now suppose that all users have packets, and the window size is set to N and only reduced by 1 after each collision. From this worst case analysis we have $T_i \leq N^2$ for all i. Hence it follows that the state space

for the Window protocol is finite, although increasing exponentially with N. Consequently, an optimal policy for the Window protocol exists and is in the class of stationary deterministic policies. Furthermore, we show in the next section that this policy exists independent of the system starting state.

3.4.3 Protocol Properties

In this section we briefly summarize some of the important properties of the Window protocol. The first four properties follow from the discussion in the previous two sections. The fifth and last property concerns the Markov chain structure for the protocol and is accompanied by a proof.

Property 1: The Window protocol is fair in the sense that the access set selected by the window always contains those users who have waited the longest since last being allowed to transmit. Note that due to the homogeneous nature of the user population, these are also the users who are most likely to have packets.

Property 2: The Window protocol has a maximum throughput of 1.0, for it is equivalent to TDMA when the window size is set to one.

Property 3: The operation of the Window protocol is at all times stable in the sense that for any $p \in [0,1]$, the average delay D satisfies

$$D \leq \text{maximum packet delay} \leq N^2$$

Property 4: The Window protocol state space is finite; hence an optimal policy exists for the infinite horizon average expected value problem and is in the class of stationary deterministic policies.

Property 5: (a) Not all stationary deterministic policies for the Window protocol correspond to indecomposable Markov chains.
(b) An optimal policy exists whose Markov chain is indecomposable.
(c) If an optimal policy has more than one irreducible set of states, then the throughput performance associated with each such set must be the same. Hence the existence of any optimal policy does not depend on the system starting state.

Proof: Part (a) may be verified, after reading Section 4.3, by considering the following policy associated with the three user problem:

(1) $w_1 = w_4 = w_7 = 3, \quad w_8 = w_{11} = w_{14} = 2, \quad w_{16} = w_{20} = w_{23} = 1$

(2) $w_2 = w_5 = 2, \quad w_{15} = w_{21} = 1$

where the window size associated with each of the remaining states is arbitrarily chosen. After examining the tree in Figure 4-5, it is clear that the states associated with part (1) of the policy and those associated with part (2) form separate irreducible classes.

For part (b) of Property 5, suppose that an optimal policy has at least two sets of irreducible states. Let $P_s(i)$ denote the system throughput given that the process starts in a state in irreducible set i, $i = 1,\ldots,m$. Also, without loss of generality, let $P_s(1) \geq P_s(2) \geq \ldots \geq P_s(m)$. Note that the Window protocol can remain in steps 2 and 3 for only a finite period of time before entering step 1 and hence a G_1 state.

Furthermore, assuming that $p < 1$, there is a nonzero probability that there will be no collisions for N consecutive slots, implying that each irreducible set of states has a G_1 state of the form

$$(\underbrace{T,\ldots,T}_{\ell_T},\underbrace{T-1,\ldots,T-1}_{\ell_{T-1}},\ldots,\underbrace{2,\ldots,2}_{\ell_2},\underbrace{1,\ldots,1}_{\ell_1})$$

where $T \in \{1,\ldots,N\}$ and $\sum_{j=1}^{T} \ell_j = N$. Now let $\ell_1,\ldots,\ell_T$ take on the values associated with such a G_1 state in irreducible set 1. For each irreducible set i, $i = 2,\ldots,m$, we choose any G_1 state, call it $S(i,T)$, and set its window size $w_{S(i,T)} = \ell_T$. Letting $S(i,T-1)$ denote the G_1 state reached from $S(i,T)$ after an empty slot or a success, we set $w_{S(i,T-1)} = \ell_{T-1}$. This is continued until $S(i,j)$, $1 \leq j \leq T-1$, is either in irreducible set 1 or is a transient state from which only set 1 states are accessible. It follows that the new policy formed by the end of this procedure will have only one irreducible set of states (that being the original class 1), and will be optimal, having a throughput of $P_s(1)$.

Part (c) also follows from the above argument. Specifically, if $P_s(1) > P_s(i)$ for any irreducible set $i \in \{2,\ldots,m\}$, then the throughput associated with starting the process in a state in set i could be increased to $P_s(1)$ through the given procedure; implying that the original policy is suboptimal.

Q.E.D.

CHAPTER 4

WINDOW PROTOCOLS FOR SMALL USER POPULATIONS

In this chapter we construct Window protocols for the user population sizes N = 2 and N = 3. For each population size we begin by first determining the system state space. Then, the associated Markov decision structure is derived and Howard's policy iteration algorithm [H5] is used to determine an optimal policy for each value of the packet generation probability p between 0 and 1. This is followed by an examination of the steady-state performance of Window protocols whose policies are based on optimizing other infinite and finite horizon objective functions. We begin by more precisely formulating the optimization problem and specifying the other objective functions that are considered.

4.1 Preliminaries

As shown in Section 3.4.2, the Window protocol state space is finite for any finite N; allowing us to restrict our search of an optimal Window protocol to the class of stationary deterministic policies. Such a policy P consists of assigning to each state S_i a window size w_i where

$$w_i \in \begin{cases} \{1,2,\ldots,N\} & \text{for } S_i \in G_1 \\ \{1,2,\ldots,\ell-1\} & \text{for } S_i \in G_2 \\ \{1,2,\ldots,\ell\} & \text{for } S_i \in G_3 \end{cases}$$

and $R = \{1,2,\ldots,\ell\}$

Let X(N) denote the Window protocol state space for the population size N and s(N) denote the number of states in X(N). We have associated with each policy $P = [w_1, w_2, \ldots, w_{s(N)}]$ a Markov chain defined on X(N) with

stationary transition probabilities $p_{ij}(w_i)$ and expected immediate rewards $r_i(w_i)$; where, given the system is in state S_i and policy P specifies window size w_i, $p_{ij}(w_i)$ is the conditional probability of moving to S_j and $r_i(w_i)$ is the conditional expected reward earned. Taking the system throughput P_s as our steady-state performance measure, we have

$$r_i(w_i) = \Pr[\text{successful transmission} \mid S_i, w_i] \tag{4.1}$$

Both $p_{ij}(w_i)$ and $r_i(w_i)$ are rational functions of p that are easily determined for any state S_i and window size w_i.

Now as indicated in Section 3.4.3, the Markov chain corresponding to a policy P may have more than one irreducible set of states. From Markov decision theory we have that the throughput performance of policy $P = [w_1, w_2, \ldots, w_{s(N)}]$, given that the system started in a state in irreducible set I (or started in any transient state from which only set I states are accessible), may be written as

$$P_s(P,I) = \sum_{i=1}^{s(N)} \pi_i(P,I) r_i(w_i) \tag{4.2}$$

where $\{\pi_i(P,I)\}$ is the stationary probability distribution of the Markov chain defined by P which satisfies

$$\pi_j(P,I) = \sum_{i=1}^{s(N)} \pi_i(P,I) p_{ij}(w_i)$$

$$1 = \sum_{i=1}^{s(N)} \pi_i(P,I)$$

and

$$\pi_i(P,I) = 0 \quad \text{for all } i \notin I$$

If the Markov chain has only one set of irreducible states (i.e., is indecomposable) or all irreducible sets have the same throughput, then the performance of policy P is independent of the system starting state. If the chain associated with P has more than one set of irreducible states with different throughputs, then it is of course desirable to start the system in that set I which maximizes (4.2) or to change the policy to ensure entrance into that set. From Property 5 of Section 3.4.3, we know that this is not a concern for any optimal Window protocol policy.

Letting $f(S_i)$ denote the number of possible window size decisions associated with state S_i, it follows that there are

$$\prod_{i=1}^{s(N)} f(S_i)$$

feasible policies to consider. With Howard's policy iteration algorithm [H5], an initial policy is selected (e.g., the policy which maximizes the expected immediate reward $r_i(w_i)$ for each state S_i), and then with each successive iteration of the algorithm a better policy is found until eventually no improvement can be made in the steady-state performance of the system for <u>any</u> starting state. A Fortran code of the policy iteration algorithm for the three user problem is given in Appendix C.

As mentioned, we are also interested in determining the steady-state performance of Window protocols whose policies are based on optimizing other infinite and finite horizon objective functions. Specifically, we examine one other infinite horizon and four finite horizon performance measures. The infinite horizon performance measure is the average rate at which the window advances along the circle, denoted by r_w. The expected immediate reward for r_w is given by

$$r_i(w_i) = (1 - \Pr[\text{a collision} \mid S_i, w_i]) \cdot w_i \qquad (4.3)$$

The performance measure r_w is not equivalent to the six steady-state performance measures defined in Section 2.1, and so any policy based on maximizing r_w will, in general, be suboptimal with respect to these other measures.

The four finite horizon performance measures are related to P_s and r_w. The first two correspond to a horizon of one slot and are simply the expected immediate rewards for P_s and r_w as defined by (4.1) and (4.3), respectively. That is, for each state $S_i \in X(N)$, the window size w_i is chosen to maximize $r_i(w_i)$. We expect, and shall see in the next two sections, that the policy associated with any reasonable Window protocol will switch to TDMA (i.e., the window size associated with the G_1 state $(N,N-1,\ldots,1)$ switches to 1) for sufficiently large p. Using the results of the Corollary that follows Theorem 2 in Appendix B, we have that the value of p at which the (4.1) criterion causes a switch to TDMA satisfies $1 - (1-p)^N = 1/2$, which implies that $p = 1 - (1/2)^{1/N}$.

For the last two performance measures, the finite horizon is that of a conflict resolution period (CRP). We define a CRP to be the interval of time between two successive entrances to step 1 of the protocol, where an empty slot or a success while at step 1 is considered a self-transition with a CRP of one slot. The two performance measures are the system throughput and average rate of window movement for one CRP. The throughput associated with a CRP which starts in G_1 state S_i is given by

$$\hat{P}_s(i) = \frac{E[\text{number of successes in CRP} \mid \text{CRP starts in } S_i]}{E[\text{duration in slots of CRP} \mid \text{CRP starts in } S_i]}$$

and similarly for the average rate of window movement we have

$$\hat{r}_w(i) = \frac{E[\text{number of users processed in CRP} \mid \text{CRP starts } S_i]}{E[\text{duration in slots of CRP} \mid \text{CRP starts in } S_i]}$$

The optimization problem is that of choosing the window size for S_i and each of the G_2 and G_3 states encountered during the CRP to maximize $\hat{P}_s(i)$ or $\hat{r}_w(i)$, as the case may be. This may be accomplished using the standard finite horizon dynamic programming algorithm [B2]. However, for N = 2 and N = 3 we shall see that the optimal window size decision for each G_2 and G_3 state remains the same for all $p \in [0,1]$. The window sizes for the G_2 and the G_3 states are set to these optimal values, leaving only the window size at the beginning of each CRP (i.e., for each G_1 state) to be determined.

In the next two sections of this chapter an optimal Window protocol and its steady-state performance are determined for the user population sizes N = 2 and N = 3, and then compared to the performance of Window protocols whose policies are derived from the above performance measures.

4.2 Two User Case

To determine the state space for N = 2, we take the system starting state to be the generic G_1 state (T_1,T_2). As illustrated in Figure 4-1, we then construct the state space X(2) by first determining which states are accessible from (T_1,T_2) for each possible window size decision and channel outcome. For any such state not in G_1, we repeat the process until each leaf of the constructed tree corresponds to a G_1 state. Note from Figure 4-1 that for any given starting state (T_1,T_2), after at most two slot-times the system will be in one of only four possible states:

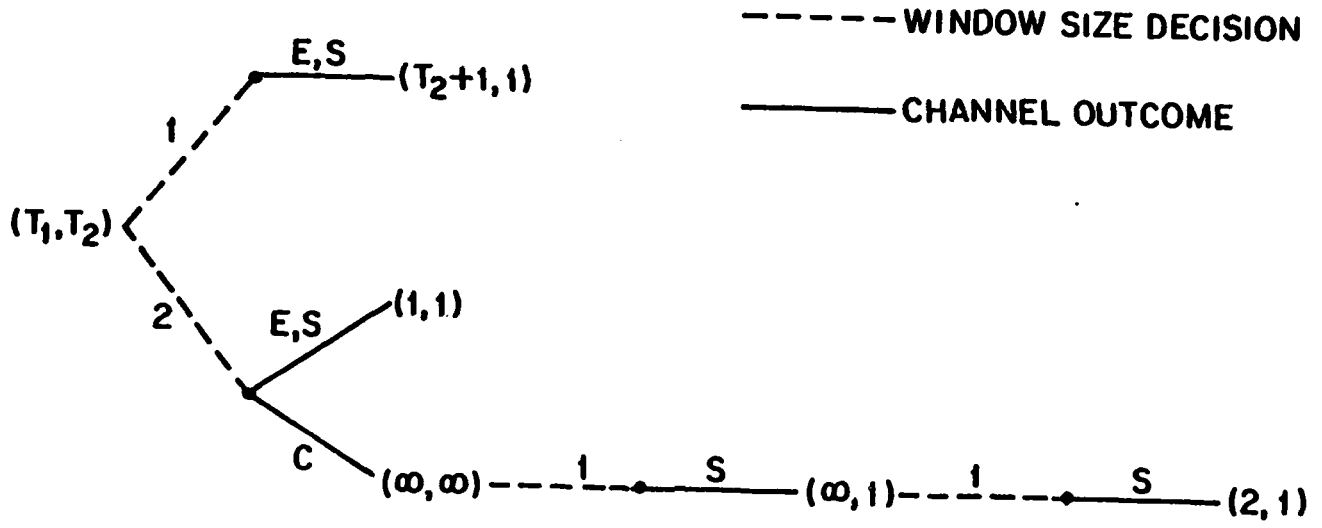


Figure 4-1 Tree of possible state transitions for N = 2

$$S_1 = (1, 1) \qquad S_2 = (\infty, \infty)$$
$$S_3 = (\infty, 1) \qquad S_4 = (2, 1)$$

These four states are taken to comprise the system state space X(2).

Note that states S_2 and S_3 correspond to, respectively, G_2 and G_3 states; and hence, as illustrated in Figure 4-1, $w_2 = w_3 = 1$. Thus we have only four feasible policies to consider:

$$P \in \{[1,1,1,1], [1,1,1,2], [2,1,1,1], [2,1,1,2]\}$$

The transition probabilities $p_{ij}(w_i)$ and the (4.1) expected immediate rewards $r_i(w_i)$ are summarized for each feasible w_i in Table 4-1. Note from either Figure 4-1 or Table 4-1 that each of the four feasible policies corresponds to an indecomposable Markov chain and, therefore, we need not be concerned about the particular starting state of the system.

4.2.1 Optimal Protocols for N = 2

Either through an exhaustive search of the four feasible policies or an application of Howard's policy iteration algorithm, an optimal policy $P^* = [w_1^*, w_2^*, w_3^*, w_4^*]$ for the two user case is found to be

$$P^* = \begin{cases} [2,1,1,2] & \text{for } 0 \leq p \leq s \\ [1,1,1,1] \text{ or } [2,1,1,1] & \text{for } s < p \leq 1 \end{cases} \tag{4.4}$$

where $s \approx 0.3473$ is the solution to $1 - 3s + s^3 = 0$ for $s \in [0,1]$. The performance of this optimal protocol as characterized by P_s is given by

$$P_s = \begin{cases} p(2 - p^2 + p^3)/(1 + p^2 + p^3) & \text{for } 0 \leq p \leq s \\ 1 - (1-p)^2 & \text{for } s < p \leq 1 \end{cases}$$

From (2.10) and (2.11) we obtain

$$D = 1 + N/P_s - 1/p$$

and thus the optimal performance in terms of the average delay D is given by

$$D = \begin{cases} 1 + p(3 + p)/(2 - p^2 + p^3) & \text{for } 0 \leq p \leq s \\ 1 + 1/(2 - p) & \text{for } s < p \leq 1 \end{cases}$$

Plotted in Figures 4-2 and 4-3 are, respectively, the D vs. p and D vs. P_s performance of the optimal protocol. Also shown is the performance of perfect scheduling for N = 2 where, following the development of

i	w_i	$p_{i1}(w_i)$	$p_{i2}(w_i)$	$p_{i3}(w_i)$	$p_{i4}(w_i)$	$r_i(w_i)$
1	1	0	0	0	1	p
	2	$1-p^2$	p^2	0	0	$2p(1-p)$
2	1	0	0	1	0	1
3	1	0	0	0	1	1
4	1	0	0	0	1	$1-(1-p)^2$
	2	1-x	x	0	0	y

$$x = p(1 - (1-p)^2) = 2p^2 - p^3$$

$$y = p(1-p)^2 + (1-p)(1 - (1-p)^2) = 3p - 5p^2 + 2p^3$$

Table 4-1 Transition probabilities and (4.1) expected immediate rewards for N = 2 state space

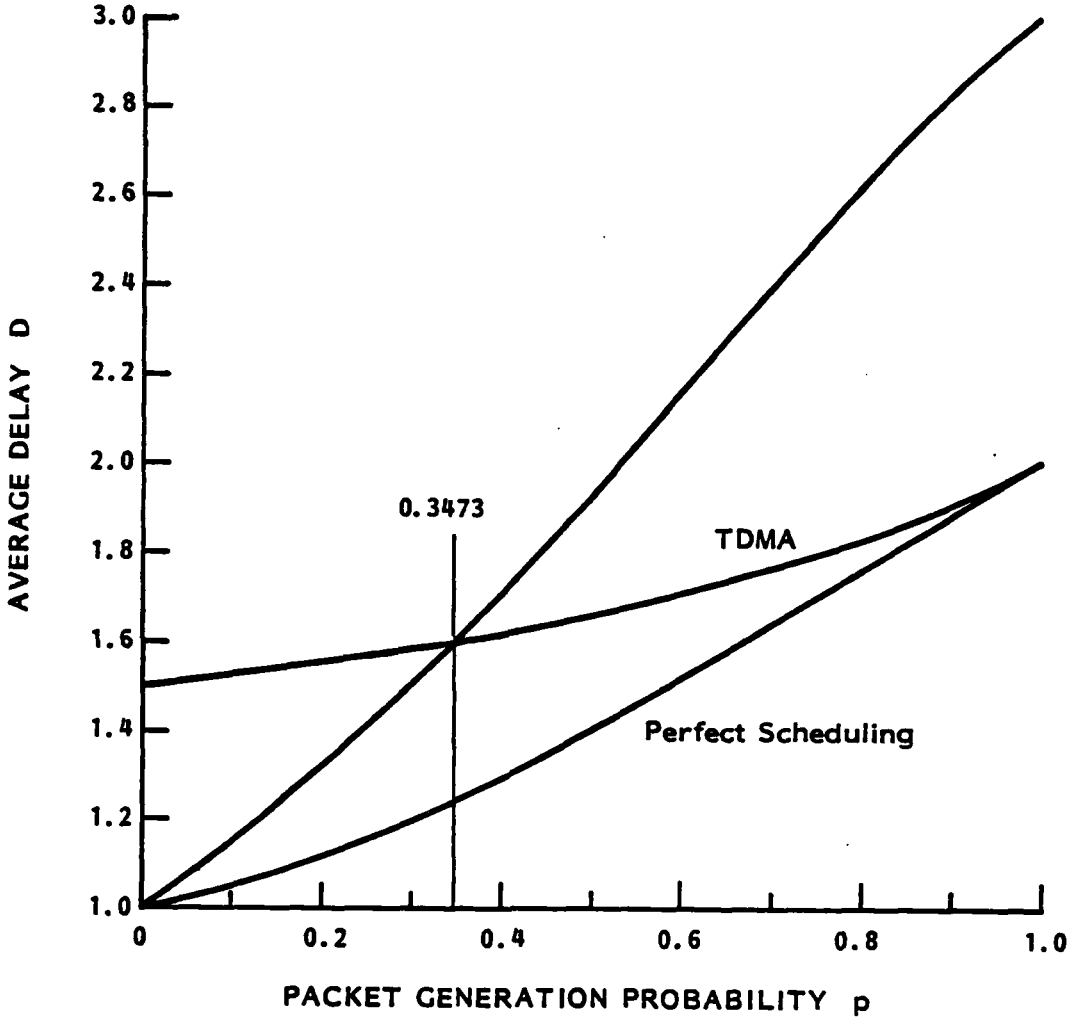


Figure 4-2 Optimal performance of Window protocol for N = 2: D vs. p

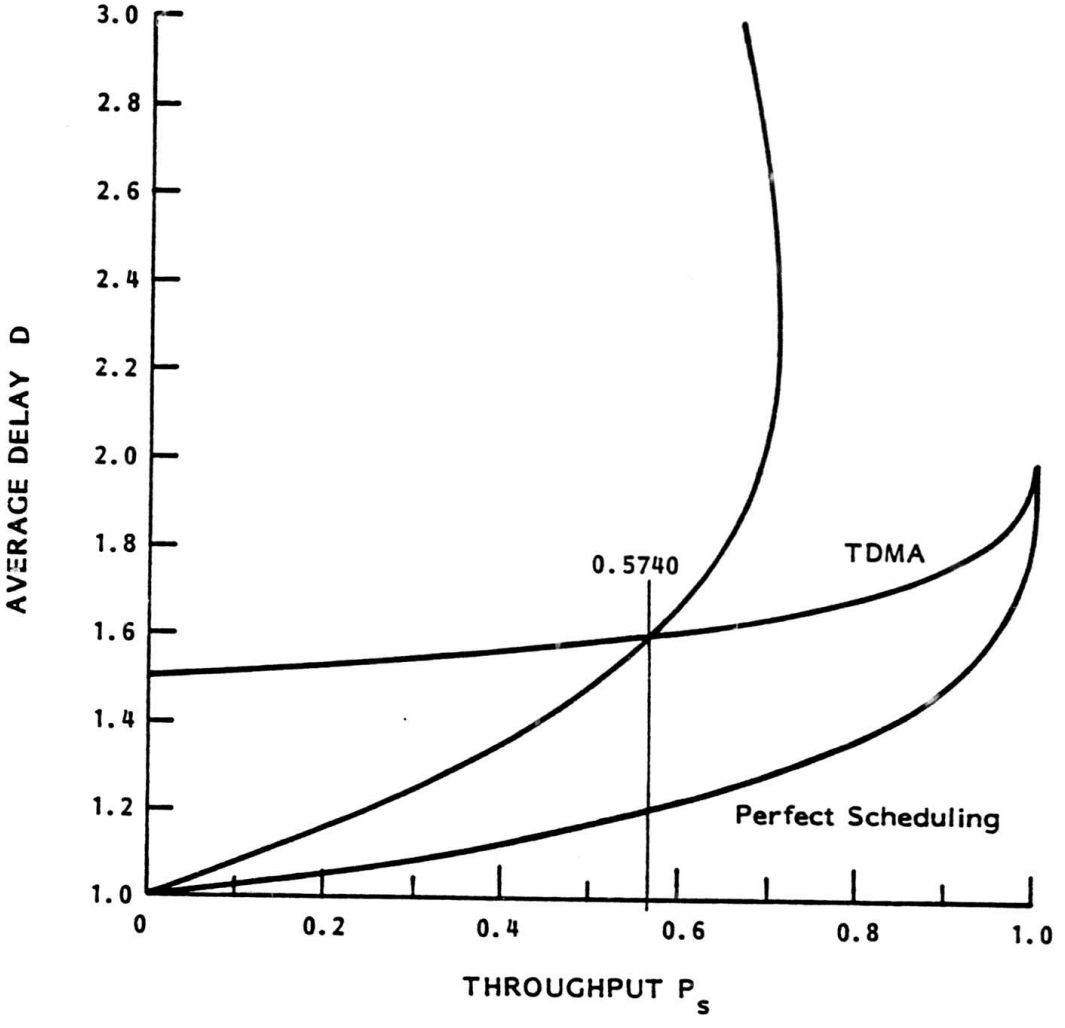


Figure 4-3 Optimal performance of Window protocol for N = 2: D vs. P_s

Section 2.2, we have

$$P_s(\text{perf. sch.}) = p(2 - 2p + p^2)/(1 - p + p^2)$$
$$D(\text{perf. sch.}) = 1 + p/(2 - 2p + p^2)$$

Note from (4.4) that for small packet generation probability p (specifically, $p \leq 0.3473$) a window size of 2 is used by the optimal Window protocol except following a collision, whereupon the window size is reduced to 1 for the next two slots, allowing each user to transmit alone. When p exceeds 0.3473 the control switches to a constant window size of 1; this, of course, is just TDMA.

Also examined for the two user case was a slight generalization of the Window protocol structure as defined in Section 3.4.1. Specifically, following the first successful packet transmission after a collision (i.e., when in S_3) we also permitted a window size of 2 (i.e., $w_3 \in \{1,2\}$). This does not change the state space X(2) and, as one might intuitively expect, results in no change in the optimal policy as given by (4.4).

4.2.2 Suboptimal Protocols for N = 2

The average delay performance of Window protocols derived by optimizing the five performance measures defined in Section 4.1 are shown in Figure 4-4, along with that of the optimal Window protocol. Note in each case that the policies given in (4.4) are also used in each of the five suboptimal protocols, the difference is the value of p at which the protocol switches from one policy to the other.

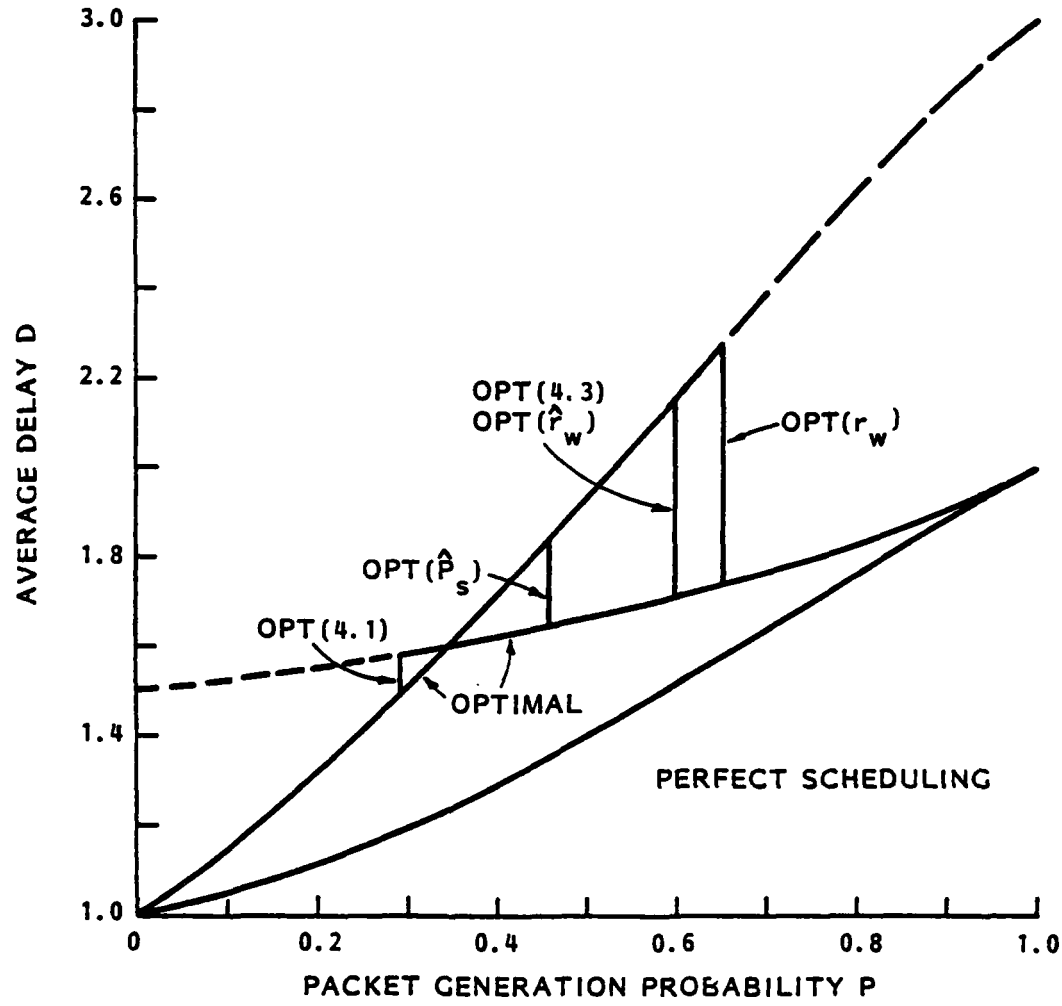


Figure 4-4 Performance of suboptimal Window protocols for N = 2

4.3 Three User Case

We construct the state space for N = 3 in the same manner used in the previous section for N = 2. Shown in Figure 4-5 is the tree of possible state transitions starting at the root with the generic G_1 state (T_1,T_2,T_3). It follows from this tree that the state space X(3) contains the 23 states listed in Table 4-2. The transition probabilities $p_{ij}(w_i)$ and expected immediate rewards $r_i(w_i)$ are easily determined for each state S_i and window size decision w_i.

4.3.1 Optimal Protocols for N = 3

A numerical application of Howard's policy iteration algorithm to this problem (see Appendix C) yields the optimal policy given in Table 4-3. Note that as the packet generation probability p varies from 0 to 1, the optimal control of the Window protocol switches among six different policies. It is easily verified that each of the six policies corresponds to an indecomposable Markov chain. The recurrent states (i.e., those belonging to the single irreducible set) for each policy are designated in Table 4-3 by a line under the corresponding window size decision. All other states for each given policy are transient. The steady-state performance of these six policies is shown in Figures 4-6 and 4-7, where the switching points are indicated by vertical lines.

There are a few important points to note concerning the window size decisions made by the optimal Window protocol given in Table 4-3. First note that if there results a collision after the window size is set to 2, the window size is reduced to 1 for two consecutive slots as in the case of N = 2. Next note that if there results a collision after the window size is set to 3, the window is reduced to size 1, allowing the first user to

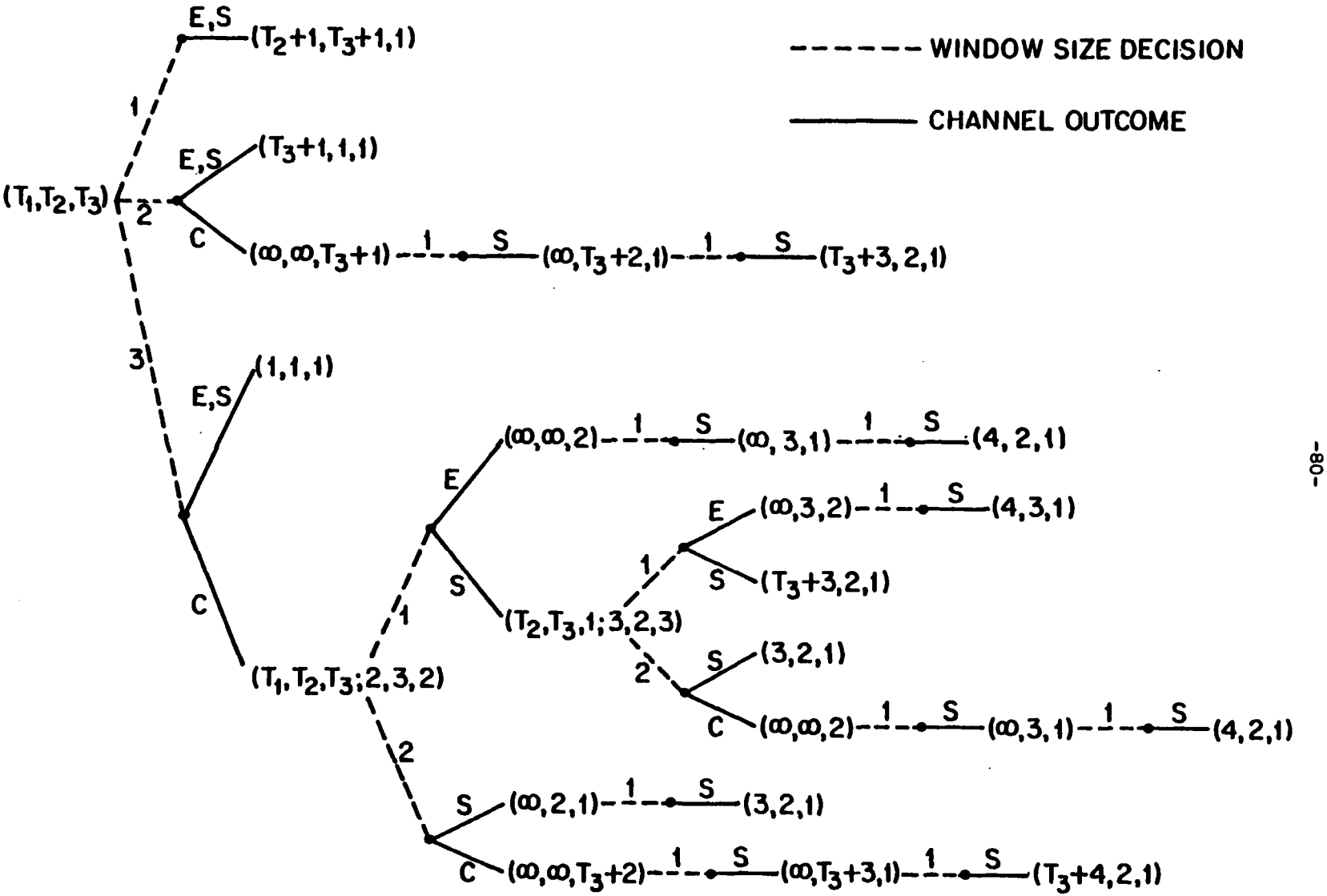


Figure 4-5 Tree of possible state transitions for N = 3

G_1	G_2	G_3
$S_1 = (1, 1, 1)$	$S_8 = (1, 1, 1; 2, 3, 2)$	$S_{17} = (1, 1, 1; 3, 2, 3)$
$S_2 = (2, 1, 1)$	$S_9 = (2, 1, 1; 2, 3, 2)$	$S_{18} = (2, 1, 1; 3, 2, 3)$
$S_3 = (2, 2, 1)$	$S_{10} = (2, 2, 1; 2, 3, 2)$	$S_{19} = (3, 1, 1; 3, 2, 3)$
$S_4 = (3, 2, 1)$	$S_{11} = (3, 2, 1; 2, 3, 2)$	$S_{20} = (\infty, 2, 1)$
$S_5 = (4, 2, 1)$	$S_{12} = (4, 2, 1; 2, 3, 2)$	$S_{21} = (\infty, 3, 1)$
$S_6 = (4, 3, 1)$	$S_{13} = (4, 3, 1; 2, 3, 2)$	$S_{22} = (\infty, 3, 2)$
$S_7 = (5, 2, 1)$	$S_{14} = (5, 2, 1; 2, 3, 2)$	$S_{23} = (\infty, 4, 1)$
	$S_{15} = (\infty, \infty, 2)$	
	$S_{16} = (\infty, \infty, 3)$	

Table 4-2 Window protocol state space for N = 3

access the channel, and then if the first user sends a packet, the window is increased to size 2 allowing the remaining two users to access the channel. This holds for all $p \in [0,1]$, so that in Figure 4-5 the window size 2 branch emanating from state $(T_1,T_2,T_3;2,3,2)$, and the window size 1 branch emanating from $(T_2,T_3,1;3,2,3)$ are never taken by the optimal protocol.

Finally, although not shown in Table 4-3, a direct application of Howard's policy iteration algorithm actually results in the optimal policy switching 17 times as p increases from 0 to 1. Specifically, the window size decision associated with each of the G_1 states switches first from 3 to 2 and then from 2 to 1, and the window size decision associated with each of the G_3 states S_{17}, S_{18}, and S_{19} switches from 2 to 1. However, only the five policy changes indicated in Table 4-3 affect the class of recurrent states and, therefore, result in an alteration of the

	state S_i																					
		G_1							G_2								G_3						
	i=1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
POLICY 1	3	3	3	3	3	3	3	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	1
POLICY 2	3	3	3	3	2	3	3	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	1
POLICY 3	3	3	3	2	2	3	3	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	1
POLICY 4	3	3	3	2	1	3	3	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	1
POLICY 5	3	2	3	2	1	3	3	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	1
POLICY 6	3	2	3	1	1	3	3	1	1	1	1	1	1	1	1	1	2	2	2	1	1	1	1

$0 \leq p \leq 0.0731$
$0.0731 < p \leq 0.0873$
$0.0873 < p \leq 0.1844$
$0.1844 < p \leq 0.2000$
$0.2000 < p \leq 0.2891$
$0.2891 < p \leq 1$

Table 4-3 Optimal Window protocol for N = 3

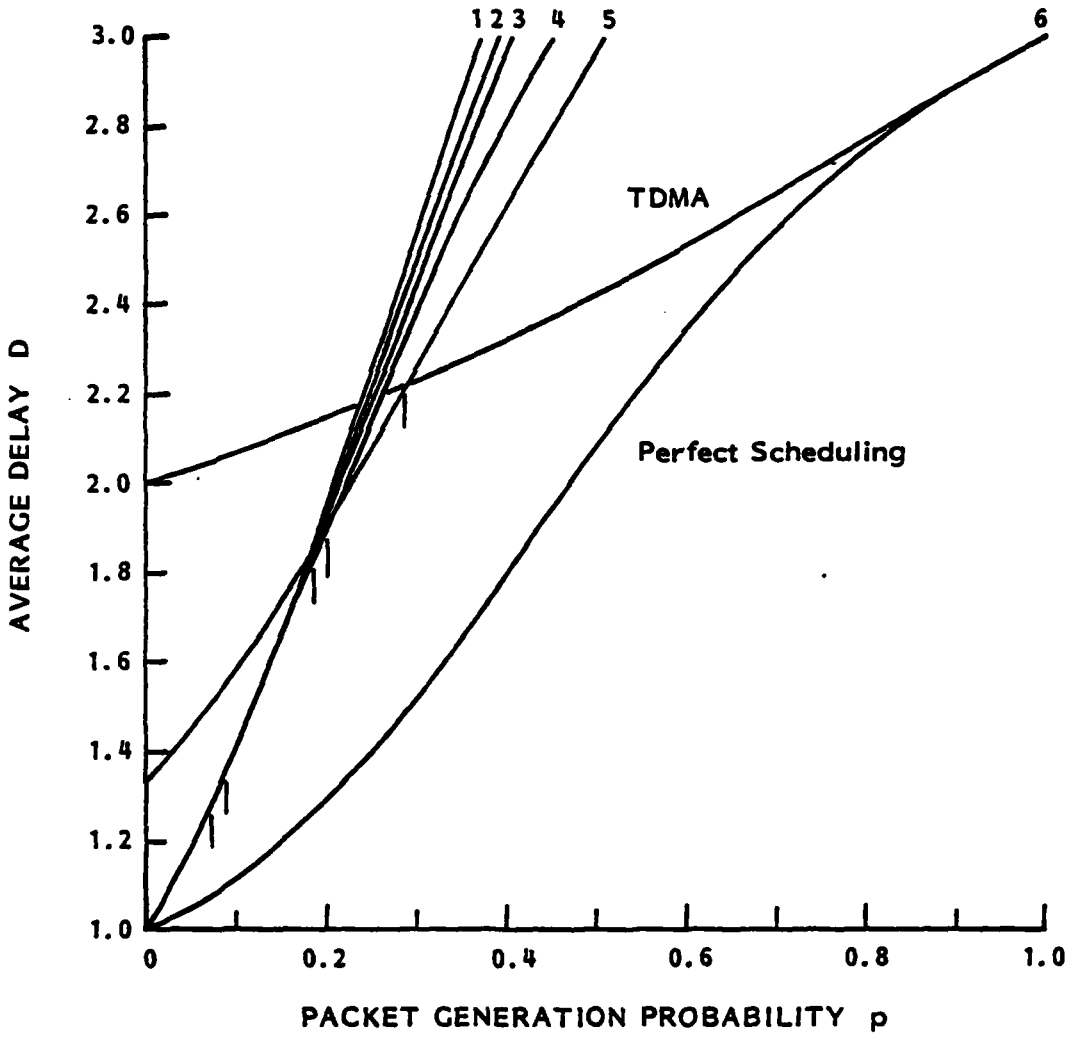


Figure 4-6 Optimal performance of Window protocol for N = 3: D vs. p

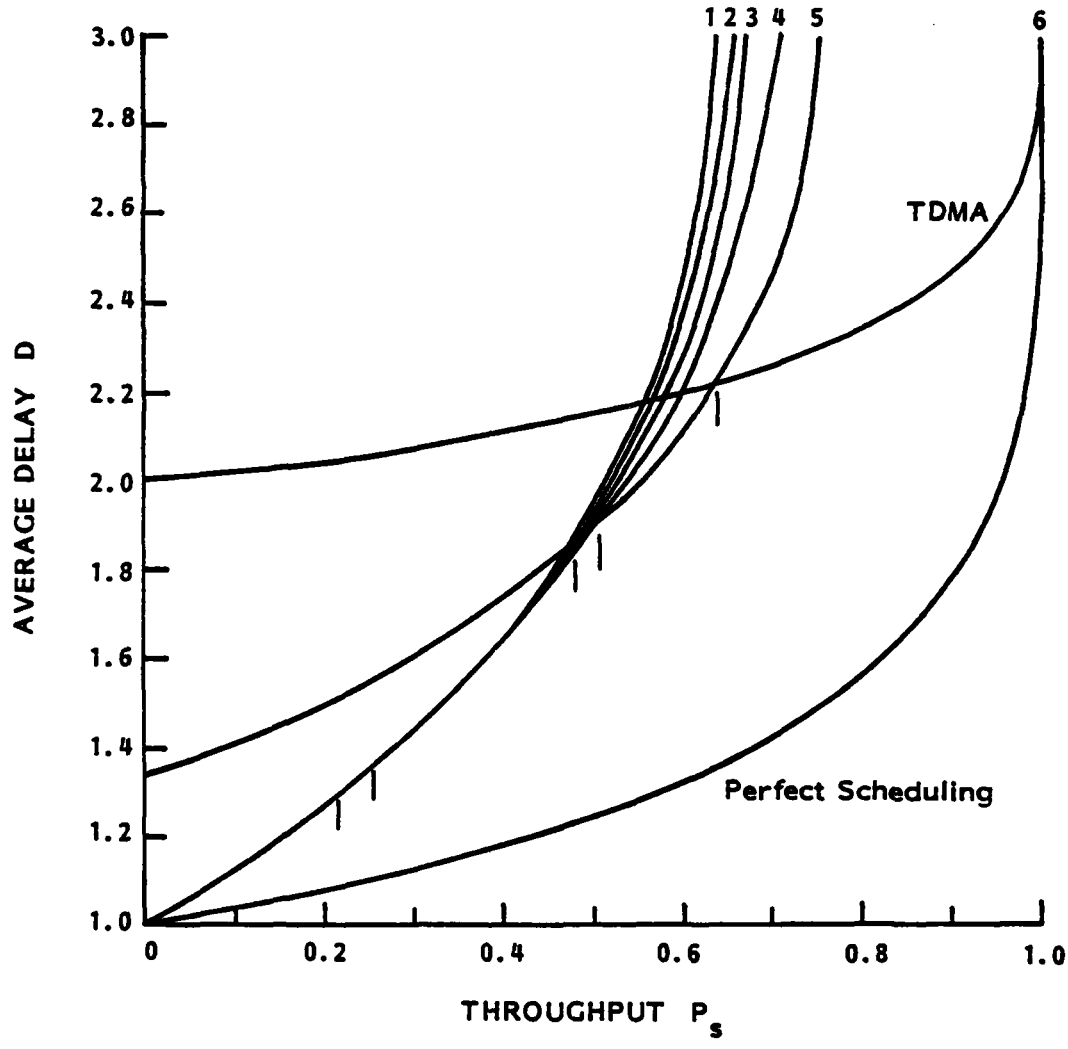


Figure 4-7 Optimal performance of Window protocol for N = 3: D vs. P_s

steady-state performance of the system. This decrease in window size as the packet generation probability p increases is illustrated in the policy changes in Table 4-3 for states S_4 and S_5. Note that this type of behavior is expected in a system that must trade off the undesirable effects of both collisions and empty slots during busy periods.

4.3.2 Suboptimal Protocols for N = 3

The average delay performance of Window protocols based on optimizing the five performance measures defined in Section 4.1 are shown in Figure 4-8, along with that of the optimal Window protocol. For the most part, each of these protocols switches among some subset of the six policies given in Table 4-3.

4.4 Four and More Users

In theory, the optimization problem for $N \geq 4$ may be handled in the same fashion as the N = 2 and N = 3 problems. The difficulty encountered is one of computational complexity: the state space increases exponentially with the population size N. For N = 4, the state space constructed in the same manner as for N = 2 and N = 3 has 223 states; an order of magnitude greater than the three user problem.

One may reduce the size of the state space by placing further restrictions on the allowable window size decisions. From the optimal Window protocols derived for N = 2 and 3, we have that the window size decisions for steps 2 and 3 are independent of p and the specific G_1 state from which step 2 was entered, depending only on the optimally selected window size for step 1 (immediately before going to step 2) and the subsequent channel outcomes while in steps 2 and 3. Hence one might

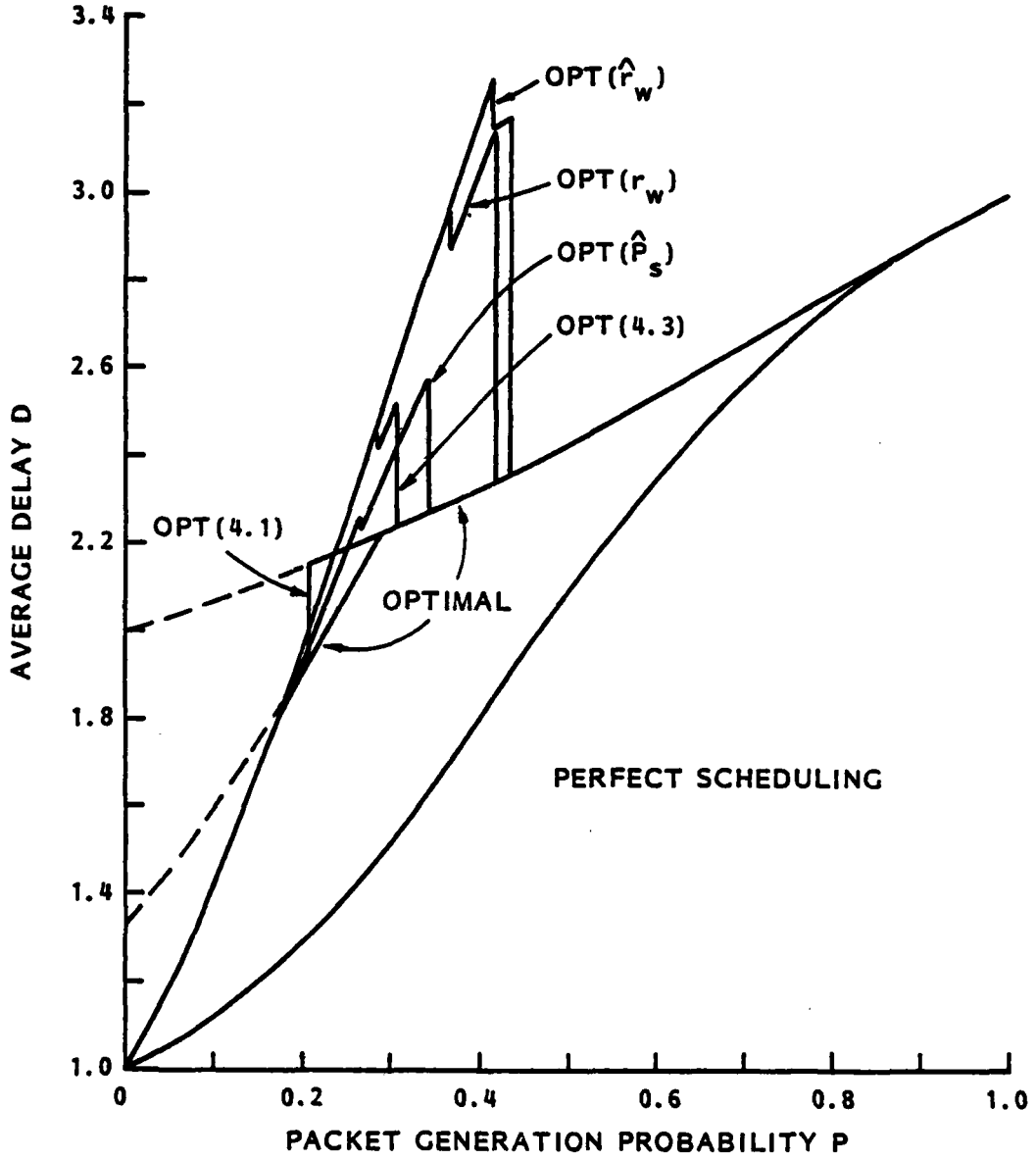


Figure 4-8 Performance of suboptimal Window protocols for N = 3

consider fixing the way in which conflicts are resolved, leaving only the window size decisions for step 1 (i.e., the G_1 states) to be determined.

It would, for example, be reasonable to simply divide the window in half after a collision, and if after doing so there is a success in the first half, to then have the window include all of the users in the second half. Doing so for $N = 4$ reduces the state space from 223 to 72 states. Given that the transition probability matrix associated with any policy is sparse (i.e., each row has at most three nonzero elements), the 72 state problem may be a reasonable one to solve. However, as we increase N further, the optimization problem quickly gets out of hand. For this reason we turn to an approximate analysis for the case of large N in the next chapter.

CHAPTER 5

WINDOW PROTOCOLS FOR LARGE USER POPULATIONS

In this chapter we consider the problem of designing Window protocols for the important case of a large user population. Of course, the immense state space characterizing this situation prevents an exact determination of an optimal policy. However, guided by what we have determined for small user populations and by what is known about infinite user populations, we construct and analyze Window protocols which perform quite well for large but finite N.

5.1 A Subclass of Window Protocols

Based on the properties of the optimal Window protocols for N = 2 and N = 3, we introduced in Section 4.4 the notion of a reasonable subclass of Window protocols. This subclass places the following two restrictions on the Window protocol structure defined in Figure 3-5: (1) the window W selected at step 2 consists of the users in the first half of the restricted class R (more precisely, the first $\lfloor |R|/2 \rfloor$ users, where $|R|$ denotes the cardinality of set R and $\lfloor x \rfloor$ denotes the largest integer less than or equal to x), and (2) at step 3, W = R. The operation of this subclass, which includes the optimal Window protocols for N = 2 and 3, is given in Figure 5-1. Note that the window size decisions for steps 2 and 3 (i.e., the conflict resolution mode) depend only on the selected window size for step 1 (immediately before going to step 2) and the subsequent channel outcomes while in steps 2 and 3. Hence, the only unspecified aspect of the protocol is that of the window size decisions for step 1. It is through the step 1 window size w that we have control over the operation

```
step 1.  W = [i,j], R = ∅
           if empty or success
             d.a. selects w ∈ {1,2,...,N}
             i ← j+1
             j ← j+w
             go to step 1

           if collision
             k ← i + ⌊(j-i+1)/2⌋
             go to step 2

step 2.  W = [i,k], R = [i,j]
           if empty
             i ← k+1
             k ← i + ⌊(j-i+1)/2⌋
             go to step 2

           if success
             i ← k+1
             go to step 3

           if collision
             j ← k
             k ← i + ⌊(j-i+1)/2⌋
             go to step 2

step 3.  W = [i,j], R = [i,j]
           if success
             d.a. selects w ∈ {1,2,...,N}
             i ← j+1
             j ← j+w
             go to step 1

           if collision
             k ← i + ⌊(j-i+1)/2⌋
             go to step 2
```

Figure 5-1 Operation of Window protocol subclass

of the protocol.

Naturally, it is important that the performance achieved by the best protocol within this subclass is at least close, if not the same, as that of the optimal protocol within the broader class defined in Figure 3-5. This we know is the case for N = 2 and 3, and intuitively should also be true for N > 3. As we have seen, designing an efficient protocol basically involves trading off the undesirable effects of collisions and empty slots during busy periods. Hence when there is a collision after selecting the window size at step 1 to avoid these two undesirable events, it is likely that only two users within the window have packets. Thus, in resolving this conflict, it is at least reasonable to initially reduce the window size by one-half and proceed as indicated in Figure 5-1. We shall see, however, that although its performance is close to optimal, halving the window size after a collision is, in fact, suboptimal for large N.

This method of dividing the conflict set in half is used by both the Tree protocol [C1,C2] and Gallager's protocol [G1] and was proposed by Kleinrock and Yemini [K7] in their extension of the Urn protocol. In fact, the restricted version of the Window protocol given in Figure 5-1 is directly analogous to the protocol devised by Gallager which is employed along the time axis for an infinite user population. The relationship between these two protocols is examined in Section 5.3.

5.2 An Analysis for Large N and Small Windows

In this section we analyze the performance and dynamic behavior of the Window protocol defined in Figure 5-1. The analysis is valid when w, the window size selected for step 1 of the protocol, is small relative to the user population size N. We begin the analysis in Section 5.2.1 by

determining the protocol performance over the duration of a single conflict resolution period (CRP). Then in Sections 5.2.2 and 5.2.3 we investigate the dynamic behavior of the protocol for, respectively, fixed and varying w.

5.2.1 CRP Performance Analysis

Recall that when at step 1 of the Window protocol, each user i independently has a packet with probability P_i given by

$$P_i = 1 - (1-p)^{T_i} \tag{5.1}$$

where p is the packet generation probability and T_i is obtained from the update rules given in Figure 3-7. For large N and a small window size w, we make the approximation

$$P_1 = P_2 = \ldots = P_w = q \tag{5.2}$$

where q is referred to as the packet occupancy probability. As illustrated in Figure 5-2, when $w << N$ the difference $T_1 - T_w$ is small relative to T_i, $i = 1,\ldots,w$, and thus (5.2) is a valid approximation. For convenience we write

$$q = 1 - (1-p)^{T} \tag{5.3}$$

where using the approximation $q = P_1$ in (5.2) we have from (5.1) that $T = T_1$.

As in Section 4.1, we define a conflict resolution period (CRP) to be the interval of time between two successive entrances to step 1 of the protocol, where an empty slot or a success while at step 1 is considered a self-transition with a CRP of one slot. Hence defining

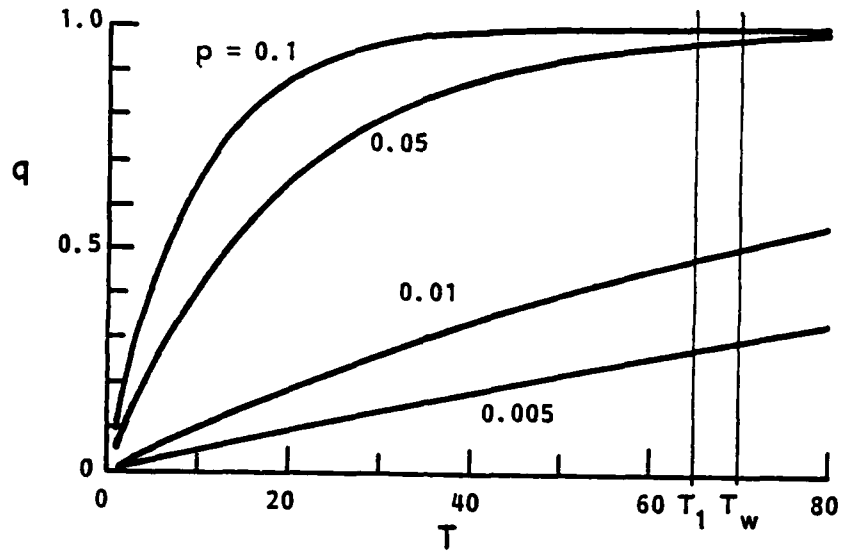


Figure 5-2 Plot of $q = 1 - (1-p)^T$ for various p

E[s] = E[number of successes in a CRP]

E[u] = E[number of users processed in a CRP]

E[t] = E[duration in slots of a CRP]

we have that $\hat{P}_s$, the system throughput over one CRP, and $\hat{r}_w$, the average rate at which the window advances along the circle during a CRP, are given by

$$\hat{P}_s = \frac{E[s]}{E[t]} \tag{5.4}$$

and

$$\hat{r}_w = \frac{E[u]}{E[t]} \tag{5.5}$$

Now let B denote the subset of w users included in the window at the beginning of a CRP but not processed during the CRP. Since each user in B independently has a packet with probability q, we have

$$E[\text{number of users in B with packets} \mid i \text{ users in B}] = iq$$

which implies that

$$E[\text{number of users in B with packets}] = q \cdot E[\text{number of users in B}]$$

Hence we obtain

$$\begin{aligned} E[s] &= E[\text{number of users processed during CRP with packets}] \\ &= qw - q\,E[\text{number of users in B}] \\ &= q \cdot E[u] \end{aligned} \tag{5.6}$$

so that from (5.4) and (5.5) we have

$$\hat{P}_s = q \cdot \hat{r}_w \tag{5.7}$$

Thus, choosing the step 1 window size w at the beginning of a CRP to maximize either $\hat{P}_s$ or $\hat{r}_w$ for that CRP are equivalent optimization problems. Note that this equivalence depends on the large N/small w analysis that we are pursuing, for we found in the small N analysis of Chapter 4 that the performance measures $\hat{P}_s$ and $\hat{r}_w$ do not generally lead to the same policy.

Proceeding with the CRP performance analysis of the Figure 5-1 Window protocol, let E[u|w] and E[t|w] denote, respectively, the dependence of E[u] and E[t] on the step 1 window size w. From Appendix D we obtain the recurrence relations

$$E[u|w] = E[u|w'] + E[u|w'']\,(e(w') + s(w')) \tag{5.8}$$

$$\begin{aligned} E[t|w] = {} & 1 - e(w')(1 + s(w'')) - e(w'')(e(w') + 2s(w')) \\ & + E[t|w'] + E[t|w'']\,(e(w') + s(w')) \end{aligned} \tag{5.9}$$

where

w' = window size for step 2 following a collision at step 1

$= \lfloor w/2 \rfloor$ *

$w'' = w - w'$

$e(w) = (1-q)^w$

and

$s(w) = wq(1-q)^{w-1}$

Hence, having the obvious result

$$E[u|1] = E[t|1] = 1$$

we may use (5.8) and (5.9) to determine, in a recursive fashion, $E[u|w]$ and $E[t|w]$ for $w = 2,3,....$ For $w = 2,3,4,5$ we obtain

$$E[u|2] = 2$$
$$E[t|2] = 1 + 2q^2$$

$$E[u|3] = 3$$
$$E[t|3] = 1 + q^2[7 - 3q]$$

$$E[u|4] = 4 - 2q^2$$
$$E[t|4] = 1 + q^2[14 - 14q + 3q^2]$$

$$E[u|5] = 5 - 3q^2$$
$$E[t|5] = 1 + q^2[25 - 35q + 16q^2 - 3q^3]$$

After determining $E[u|w]$ and $E[t|w]$ we may, using Equations (5.5)-(5.7), obtain expressions for $\hat{r}_w$, $E[s|w]$, and $\hat{P}_s$. For $w = 1,...,5$, Figure 5-3 shows the CRP throughput $\hat{P}_s$ vs. packet occupancy probability q

---

* The recurrence relations (5.8) and (5.9) do not depend on $w' = \lfloor w/2 \rfloor$.

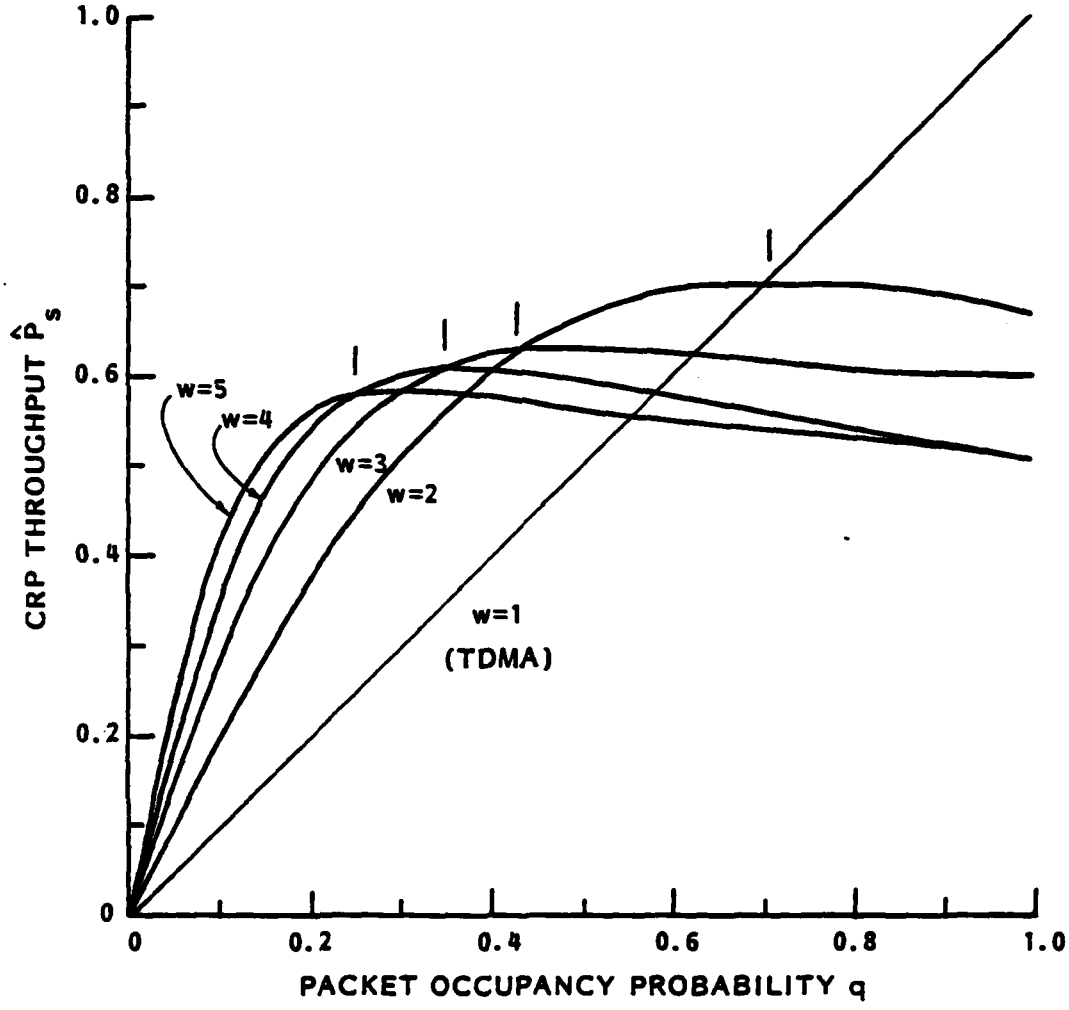


Figure 5-3 Window protocol performance: $\hat{P}_s$ vs. q

performance of the protocol. The switching points for the optimal window size w^* are indicated in Figure 5-3 by vertical lines and are also listed in Table 5-1 for $0.06038 \leq q \leq 1.0$. Note from Table 5-1 that window sizes 14 and 18 are not used in the optimal finite horizon (i.e., one CRP) control of the protocol. That is, w^* switches directly from 13 to 15 and from 17 to 19. This occurs not because there is something fundamentally wrong with the step 1 window sizes 14 and 18, but rather because dividing the conflict set in half is generally suboptimal. If we were to take $w' = 6$ and 8 for $w = 14$ and 18, respectively, then w^* would switch through 14 and 18 as we decrease q from 1.0. However, the improvement in performance is minimal ($\hat{P}_s$ increases on the order of 5×10^{-4}). The analogous situation for the infinite population problem is discussed in Section 5.3.

5.2.2 Dynamic Analysis for Fixed w

In this section, under the assumption of a fixed step 1 window size w, we examine the dynamic behavior of the packet occupancy probability q. Of course if $w = 1$, then q is a constant since $T = N$ in (5.3). Hence, the interesting case is when $w \geq 2$.

For a given w, packet generation probability p, and user population size N, we define the desired equilibrium operating point of the protocol to be the smallest value of $q_o \in [0,1]$ satisfying

$$q_o = 1 - (1-p)^{T_o} \tag{5.10}$$

where

$$T_o \triangleq \frac{N}{\hat{F}_w(q_o)} \tag{5.11}$$

w^*	q
1	1.0000
1 — 2	0.7071
2 — 3	0.4294
3 — 4	0.3534
4 — 5	0.2472
5 — 6	0.2130
6 — 7	0.1979
7 — 8	0.1691
8 — 9	0.1373
9 — 10	0.1211
10 — 11	0.1196
11 — 12	0.1051
12 — 13	0.1047
13 — 15	0.09183
15 — 16	0.08092
16 — 17	0.07423
17 — 19	0.06681
19 — 20	0.06038

Table 5-1 Step 1 window size switching points for the optimal finite horizon control of the (Fig. 5-1) Window protocol

Note that T_o represents the expected time for a complete revolution of the window about the circle if q is held fixed at q_o. Substituting (5.11) into (5.10) yields

$$-\hat{r}_w(q_o)\ln(1-q_o) = -N\ln(1-p) \tag{5.12}$$

Since for $0 \leq q_o \leq 1$, $\hat{r}_w(q_o)$ remains bounded and $-\ln(1-q_o)$ ranges from 0 to ∞, we have from (5.12) that (5.10) is satisfied for at least one value of $q_o \in [0,1]$. From (5.3) and (5.10) we may write

$$q = 1 - (1-q_o)^{T/T_o} \tag{5.13}$$

We wish to determine under what circumstances q will remain, in a statistical sense, near the equilibrium point q_o.

Initially, let us consider the case where w = 2. We assume for simplicity that N is even, and we take the packet occupancy probability associated with the protocol for the first revolution of the window about the circle to be q_o. At the start of the second revolution, the new value for q is given by (5.13) where

$$T = \sum_{i=1}^{N/2} t_i \tag{5.14}$$

and the random variables t_i, $i = 1,\ldots,N/2$, are independent and identically distributed with

$$t_i = \begin{cases} 3 & \text{with prob. } q_o^2 \\ 1 & \text{with prob. } 1 - q_o^2 \end{cases}$$

It follows that

$$E[T] = T_o = N(1 + 2q_o^2)/2$$

and the variance

$$\sigma^2(T) = N2q_o^2(1 - q_o^2)$$

Hence we may write

$$T = T_o + O(N) \tag{5.15}$$

where O(N) is a zero mean random variable with standard deviation $\sqrt{N2q_o^2(1-q_o^2)}$. Substituting (5.15) into (5.13) yields

$$q = 1 - (1-q_o)^{1 + O(N)/T_o} \tag{5.16}$$

By the weak law of large numbers [F2], we have that for any $\varepsilon > 0$, $Pr[O(N)/T_o > \varepsilon] \to 0$ as $N \to \infty$; and thus from (5.16), $q \to q_o$ in a like manner. Hence, for sufficiently large N, after the first revolution of the window, q will remain close in value to q_o with high probability.

To make this concept more quantitative, consider the case where w = 2 and $q_o = 0.5$. From (5.13) we obtain

$$T = T_o \ln(1-q)/\ln(1-q_o) \tag{5.17}$$

Letting n_c denote the number of conflicts in the first complete cycle of the window, we have

$$T = N/2 + 2n_c$$

and thus

$$\begin{aligned} n_c &= T/2 - N/4 \\ &= T_o \ln(1-q)/2\ln(1-q_o) - N/4 \\ &= N[\gamma \ln(1-q) - 1/4] \end{aligned} \tag{5.18}$$

where $\gamma \triangleq T_o/2\ln(1-q_o) \approx -0.5410$ for $q_o = 0.5$. We wish to determine α, the probability that $|q-q_o| \leq 0.05$, for various N. We have

$$\begin{aligned} \alpha &= \Pr[0.45 \leq q \leq 0.55] \\ &\approx \Pr[0.0734N \leq n_o \leq 0.1820N] \qquad (5.19) \\ &\approx Q[(-0.0516N - 0.5)/\ 0.09375N] \\ &\qquad - Q[(0.0570N + 0.5)/\ 0.09375N] \qquad (5.20) \end{aligned}$$

where (5.19) follows from (5.18), and (5.20) follows from an application of the DeMoivre-Laplace limit theorem [F2] with Q-function defined by

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-y^2/2}\, dy$$

Table 5-2 lists computed values of α for various N.

An analysis similar to the above may be used to show that $q \to q_o$ in probability as $N \to \infty$ for any finite w. However, the expected number of independent random variables t_i in (5.14) generally decreases with increasing w. Thus a given confidence interval will require successively

N	α
50	0.8616
100	0.9462
150	0.9780
200	0.9907
250	0.9960
300	0.9982

Table 5-2 α for various values of N

larger values of N as we increase w.

Now although we have shown that after the first cycle q will remain near the equilibrium value q_o (with high probability for sufficiently large N), this does not ensure that q will remain close to q_o over the infinite horizon operation of the protocol. It is conceivable that q could gradually drift away from q_o. As we shall see, this does not happen for $w < 11$, but can happen for $w \geq 11$.

Suppose at some time in the operation of the protocol that $q = q' \neq q_o$. As illustrated in Figure 5-4, q' is determined from (5.13) by a unique value of T, say T'. We define the system drift at q' to be

$$d(q') = \bar{T}(q') - T' \tag{5.21}$$

where

$$\bar{T}(q) \triangleq \frac{N}{\hat{r}_w(q)} = T_o \quad \text{for } q = q_o \tag{5.22}$$

That is, if q is held fixed at q' for a complete cycle, then d(q') represents the expected change in T at the end of the cycle. Of course, q does not remain fixed; but by dividing d(q') by N we obtain the drift per processed user which is valid for at least the beginning of the cycle.

We wish to determine under what conditions

(1) $d(q) < 0 \quad \text{for} \quad q > q_o$

and

(2) $d(q) > 0 \quad \text{for} \quad q < q_o$

From (5.12), (5.17), (5.21), and (5.22) we may rewrite (1) and (2) as

(1) $-\hat{r}_w(q)\ln(1-q) > -N\ln(1-p) \quad \text{for} \quad q > q_o$

and

(2) $-\hat{r}_w(q)\ln(1-q) < -N\ln(1-p) \quad \text{for} \quad q < q_o$

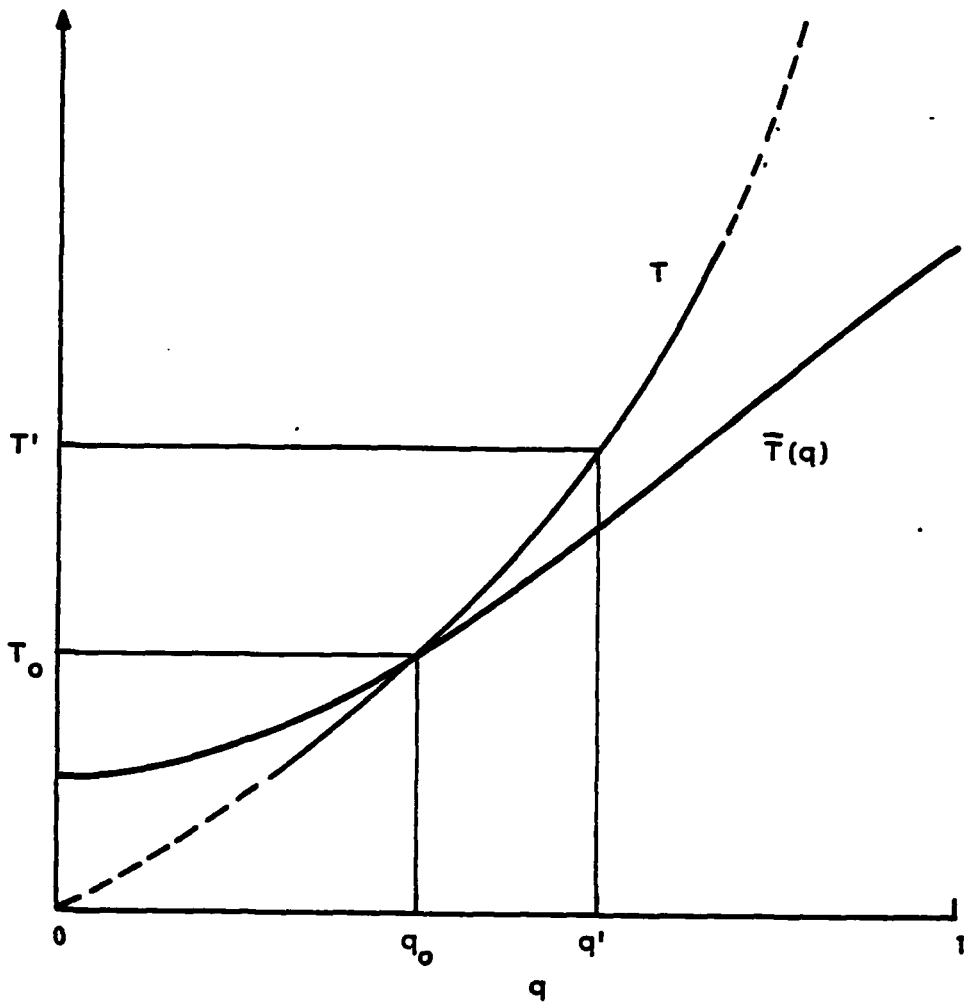


Figure 5-4 Plot of T and $\bar{T}(q)$ vs. q

Hence it follows that (1) and (2) hold if and only if (5.10) is satisfied for exactly one value of q_o. A sufficient condition for this desirable drift property is that

$$\frac{\partial}{\partial q}\{-\hat{r}_w(q)\ln(1-q)\} > 0 \qquad (5.23)$$

for all $q \in [0,1]$. As shown in Figure 5-5, (5.23) is satisfied for $w = 1,\ldots,5$. However, as we continue to increase w, $-\hat{r}_w(q)\ln(1-q)$ will eventually form two stationary points. This occurs when $w \geq 11$, and is illustrated in Figure 5-6 for $w = 16$. Note that, for the value of $-N\ln(1-p)$ shown, there are three equilibrium points. The first (at $q = q_o$) is the desired operating point of the system and is stable in the sense that small excursions result in statistical drifts tending to restore the equilibrium. The second is unstable and the third is stable but undesirable. Statistical fluctuations will cause the system to oscillate between the two stable points. Of course this bistable behavior can be avoided by sufficiently increasing or decreasing $-N\ln(1-p)$, or by changing the step 1 window size w. However, the value of q_o shown in Figure 5-6 is in the range for which $w = 16$ maximizes $\hat{P}_s$ (see Table 5-1). This indicates that there is an inherent bistability of the protocol when q_o is small and the corresponding (large) value of w is selected to optimize the system performance at this point. However, as we discuss in the next section, by dynamically varying w with q, one can force q to drift back to q_o.

As indicated, for $w < 11$ we have that if q drifts from the equilibrium point q_o, then even without changing the step 1 window size w, q is expected (in the probabilistic sense) to move back toward q_o. Thus one can envision q fluctuating about q_o where, from the law of large numbers result, the range of fluctuation decreases with increasing N. In fact, if

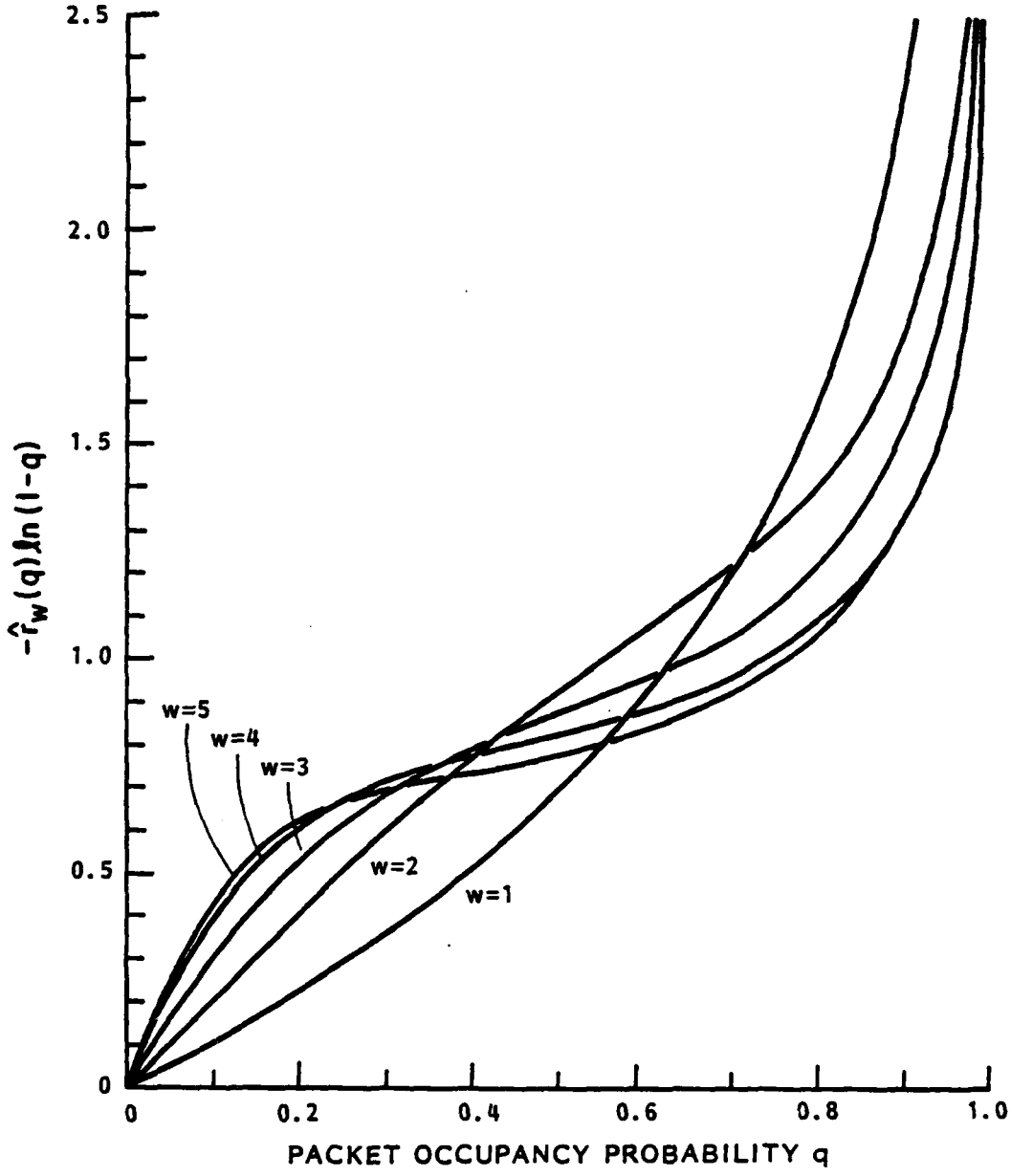


Figure 5-5 Plot of $-\hat{r}_w(q)\ln(1-q)$ vs. q for w = 1,...,5

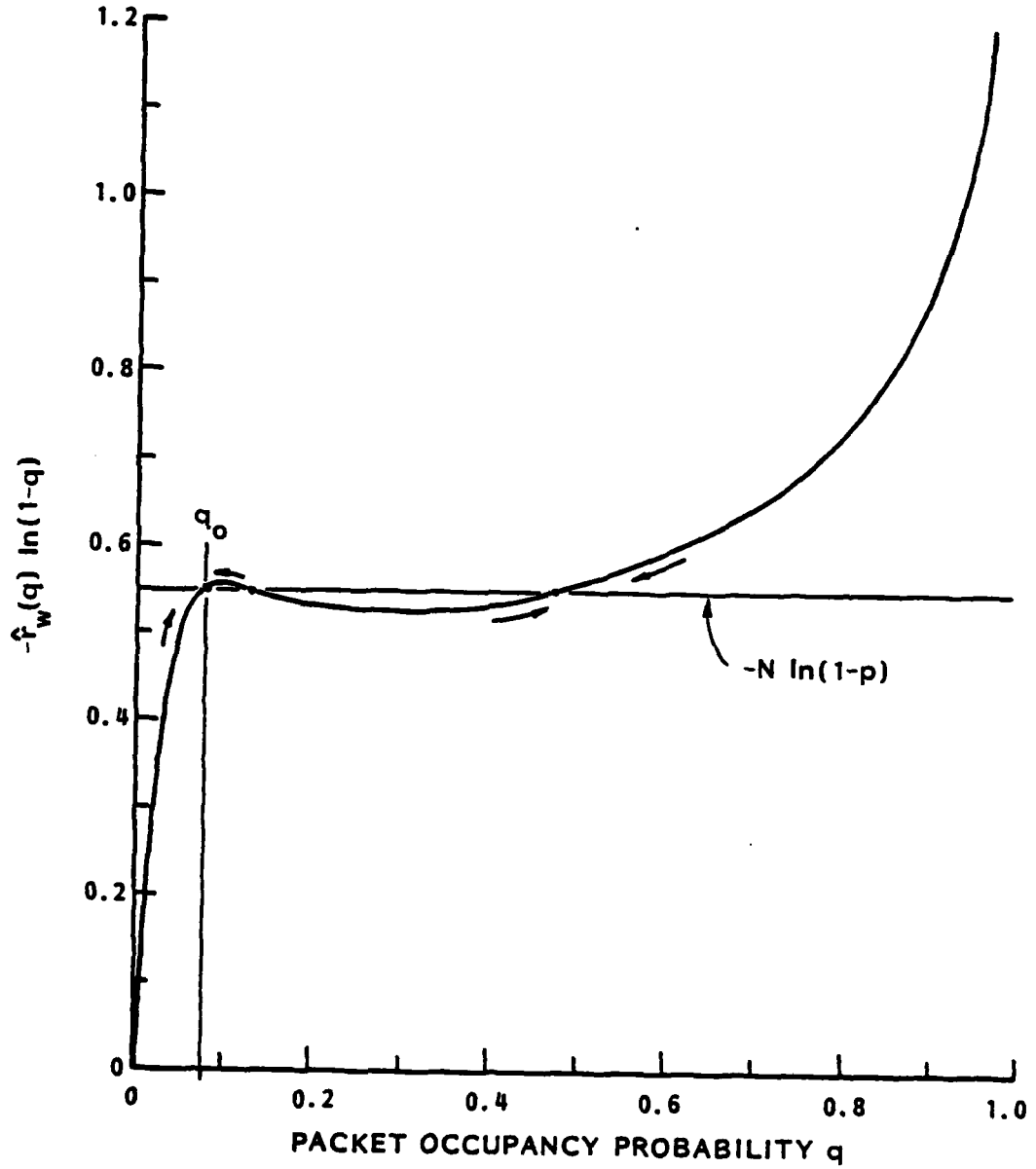


Figure 5-6 Plot of $-\hat{r}_w(q)\ln(1-q)$ vs. q for w = 16

N is sufficiently large that q remains relatively constant over the infinite horizon operation of the protocol, then the CRP performance measures $\hat{P}_s$ and $\hat{r}_w$ are equivalent to their infinite horizon counterparts P_s and r_w, respectively. Moreover, from (5.7) we have that P_s and r_w are simply related, and from the results of Section 2.1, are in turn directly related to the steady-state performance measures B_u, P_b, B_a, N_s, and D. To get an idea of the Window protocol's steady-state performance in this situation, Figure 5-7 shows for N = 50 the average delay D vs. throughput P_s performance for w = 1,...,5. Each curve is, in essence, parameterized in q, where D is determined from Equation (5.25) in Section 5.3. Note from Figure 5-7 that as we allow the largest feasible window size w to increase, the incremental improvement in the protocol's optimal performance decreases.

5.2.3 Dynamic Analysis for Varying w

We now examine the dynamic behavior of the Window protocol when w is allowed to vary with q. That is, at the beginning of each CRP, q is determined from (5.3) and then w is selected using the switching points given in Table 5-1. For given values of p and N, the desired equilibrium operating point of the system is defined to be the minimum $q_o \in [0,1]$, over all $w \in \{1,2,...\}$, which satisfies (5.10). Graphically, q_o and w are found by following along the line -Nln(1-p) in Figure 5-5. Beginning at q = 0, the first curve intersected by this line as q is increased determines w and the point of intersection determines q_o.

Of course it makes intuitive sense to have w vary dynamically with q. The idea is that if during a cycle of the window there are many collisions so that $T > T_o$, implying $q > q_o$, then it might be desirable to temporarily

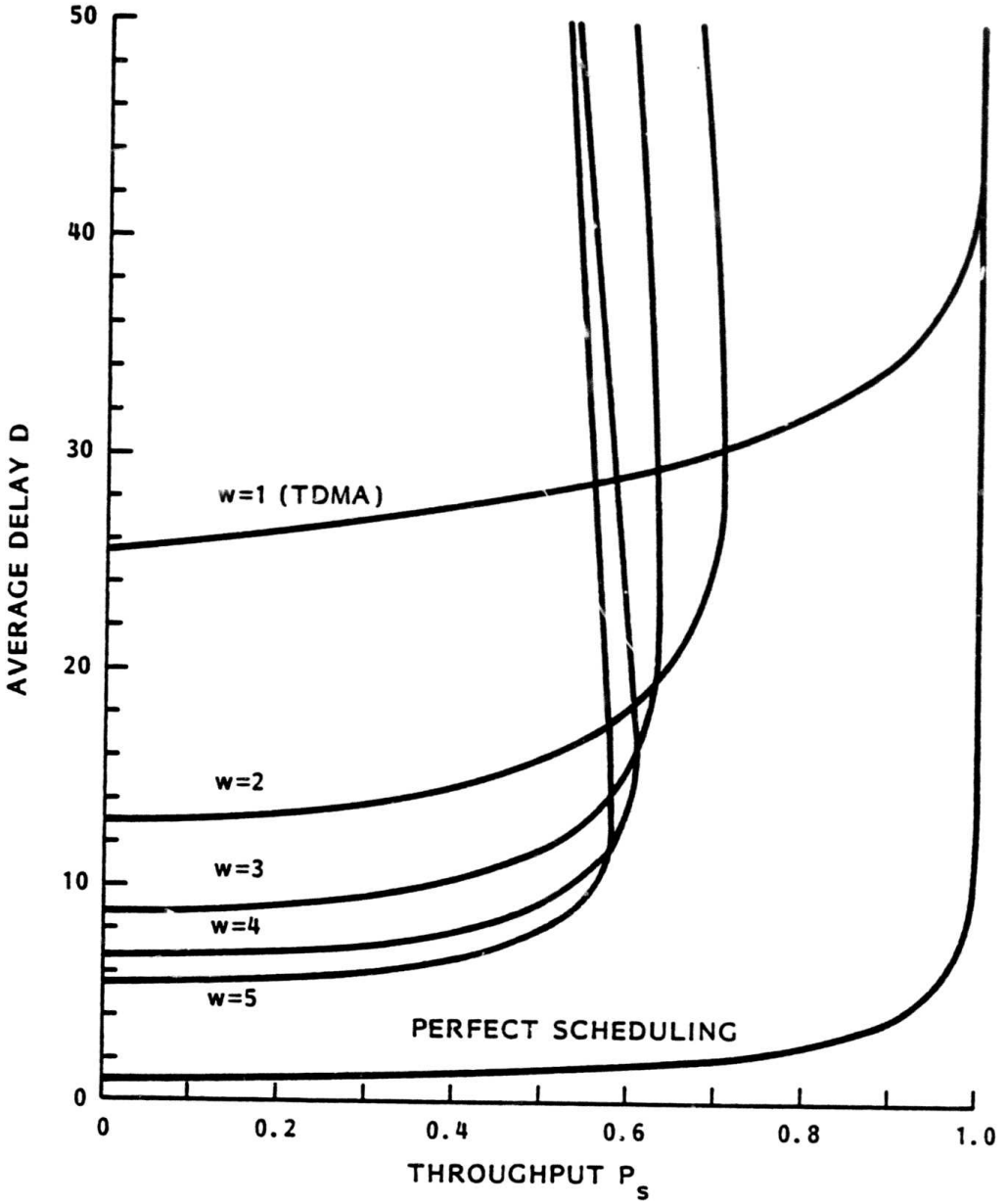


Figure 5-7 Window protocol performance for N = 50: D vs. P_s

reduce w from the value associated with q_o to better balance the collision vs. empty slot trade-off. Similarly, if there are many empty slots during a cycle so that $T < T_o$, and thus $q < q_o$, then it might be desirable to increase w. This is, of course, precisely how the optimal CRP control of the Window protocol (as specified, e.g., in Table 5-1) behaves. Moreover, we expect the optimal infinite horizon control (i.e., that which maximizes the steady-state throughput P_s) to be of the same form. The two controls should only differ in the values of q at which the optimal step 1 window size w^* changes, and possibly which values of w are not included in the optimal control.

Let us assume that the window size w varies with q according to the optimal CRP control of the protocol. The result we wish to establish is that under this control, the drift of q is always toward the desired equilibrium point q_o. Following the development in the previous section, it suffices to show that for any $q \in [0,1]$, (5.23) is satisfied for that w specified for q by the control. In other words, the upper envelope for the set of curves $-\hat{r}_w(q)\ln(1-q)$ vs. q, w = 1,2,..., is monotonically increasing, and therefore can only intersect the line $-N\ln(1-p)$ once. The cases w = 1,...,20 were examined numerically, and this condition was verified. The verification was not pursued beyond w = 20. Note, however, that an equivalent condition for this single equilibrium point is that

$$\tilde{P}_s(q) = \max_w \hat{P}_s(q)$$

(i.e., the upper envelope for $\hat{P}_s(q)$, w = 1,2,...) intersects $qN\ln(1-p)/\ln(1-q)$ once in the interval $q \in [0,1]$. Since $qN\ln(1-p)/\ln(1-q)$ is monotonically decreasing over $q \in [0,1]$, it suffices to show that $\tilde{P}_s(q)$ is monotonically increasing in this interval. This is the case for

$0.06038 \leq q \leq 1$, and from the maximum throughput results in Table 5-3 (where $w = 2^k$ and $P_s(q^*)$ is shown to decrease monotonically as $k \rightarrow \infty$), we expect that it will also be true for $q < 0.06038$. This is illustrated graphically in Figure 5-8 for $0 \leq q \leq 0.2$ and $q_o = 0.08$. Also shown is the expected rate at which q converges to q_o from some point q'. Specifically, for large N, we have that q", the expected value for q at the start of the next revolution of the window, is given by

$$q'' = 1 - (1-p)^{N/\tilde{r}_w(q')}$$

where $\tilde{r}_w(q) \triangleq$ maximum $\hat{r}_w(q)$ over all w. This implies that

$$\frac{\tilde{P}_s(q')}{q'} = \frac{N\ln(1-p)}{\ln(1-q'')}$$

Hence it follows that the point at which line ℓ in Figure 5-8 intersects $qN\ln(1-p)/\ln(1-q)$ determines q". Thus, the expected convergence of q to q_o is as illustrated in the figure.

Note that in selecting w at the beginning of a CRP according to the current value of q, there is no direct dependency of the control on N or p. Moreover, it is not even necessary to know the equilibrium operating point of the system q_o.

5.3 Relationship Between the Finite and Infinite User Population Problems

In this section we investigate the behavior of the Figure 5-1 Window protocol as we let $N \rightarrow \infty$. The results obtained help establish a link between the finite and infinite user population problems which thus far have largely been treated separately.

We begin by examining the performance associated with successively larger values of w. We assume that N is finite but as large as needed to

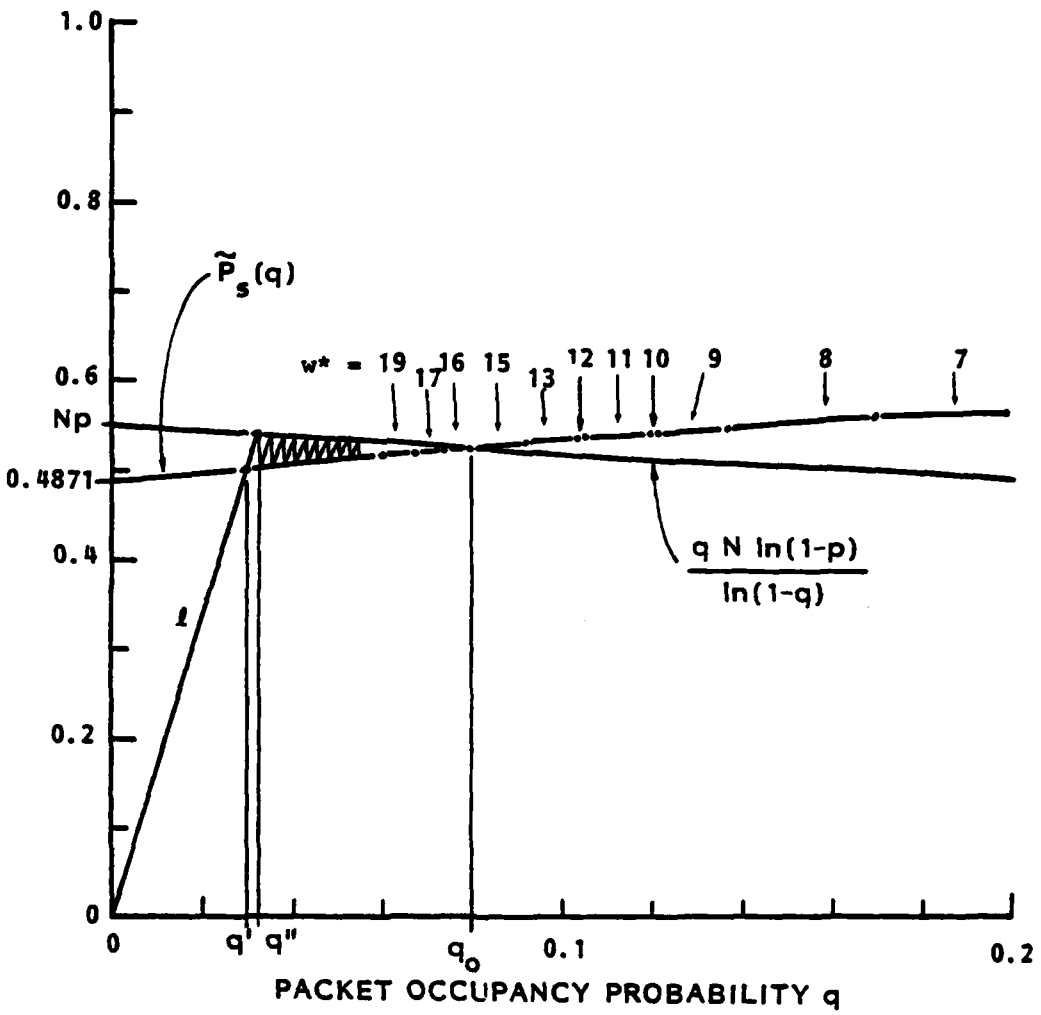


Figure 5-8 Illustration of system stability for varying w where q_o = 0.08

make the performance analysis valid (i.e., $q = q_o$ and thus $P_s = \hat{P}_s$) for the particular w being considered.* To allow us to examine very large w without incurring a computational burden, we restrict our attention to $w = 2^k$, $k = 0,1,2,....$ The recursion given in Appendix D simplifies since $w' = w'' = 2^{k-1}$ for $w = 2^k$, $k = 1,2,....$ The throughput performance for $k = 0,...,10$ is given in Figure 5-9. Letting q^* denote the value of q which maximizes the throughput P_s for a given k, and q_s denote the switching point from one value of k to the next, Table 5-3 lists q^*, $P_s(q^*)$, q^*w, q_s, and $P_s(q_s)$ for $k = 0,...,20$. Note that qw is equal to the expected number of packets held by users within the window of size w. Observe from Table 5-3 that as $k \to \infty$, $P_s(q^*) \to 0.4871$ and $q^*w \to 1.266$. These are precisely the results obtained by Gallager [G1] in the maximum throughput analysis of his protocol for an infinite user population. Even though Gallager's protocol operates along the time axis and our's along the "user circle", in the limit as N and $w \to \infty$ and $q \to 0$, the two are statistically equivalent.

In an extension to Gallager's work, Mosely [M8] found that dividing the conflict set in half, although close to being optimal, is in fact suboptimal. She determined that the maximum throughput could be increased to 0.4877 by using a slightly larger initial time interval (i.e., one equivalent to $qw = 1.275$) and then, if there occurred a collision, reduce the time interval to 0.465 (rather than 0.5) of its initial value. On subsequent reductions of the time interval due to conflicts within the same CRP, this factor approaches 0.5. In addition, she found that when a

---

* Strictly speaking, to maintain the stability of the system, w will have to varied, as indicated in the previous section, over the infinite horizon operation of the protocol. However, by taking N sufficiently large, needed changes in w may be made as rare as desired.

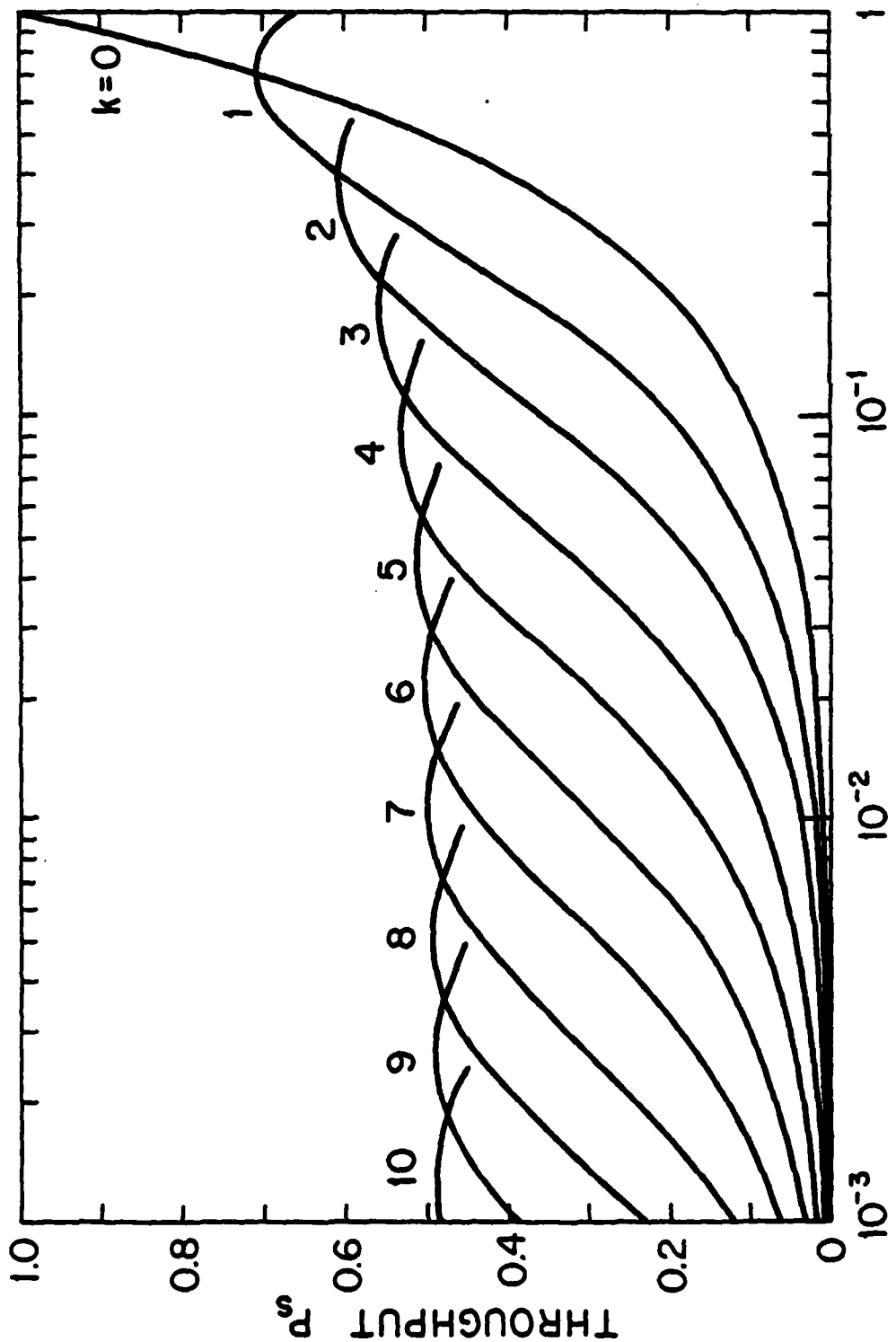


Figure 5-9 Throughput performance of Window protocol for $w = 2^k$

k	w	q^*	$P_s(q^*)$	q^*w	q_s	$P_s(q_s)$
0	1	1.0000	1.0000	1.000		
					0.7071	0.7071
1	2	0.7071	0.7071	1.414		
					0.4024	0.6079
2	4	0.3701	0.6090	1.480		
					0.2146	0.5544
3	8	0.1809	0.5579	1.447		
					0.1108	0.5224
4	16	0.8712×10^{-1}	0.5285	1.394		
					0.5633×10^{-1}	0.5028
5	32	0.4217×10^{-1}	0.5113	1.349		
					0.2840×10^{-1}	0.4909
6	64	0.2060×10^{-1}	0.5011	1.318		
					0.1426×10^{-1}	0.4837
7	128	0.1014×10^{-1}	0.4951	1.298		
					0.7143×10^{-2}	0.4795
8	256	0.5019×10^{-2}	0.4917	1.285		
					0.3575×10^{-2}	0.4770
9	512	0.2494×10^{-2}	0.4897	1.277		
					0.1788×10^{-2}	0.4756
10	1024	0.1243×10^{-2}	0.4885	1.272		
					0.8944×10^{-3}	0.4747
11	2048	0.6200×10^{-3}	0.4879	1.270		
					0.4473×10^{-3}	0.4743
12	4096	0.3096×10^{-3}	0.4875	1.268		
					0.2237×10^{-3}	0.4740
13	8192	0.1547×10^{-3}	0.4873	1.267		
					0.1118×10^{-3}	0.4739
14	16384	0.7733×10^{-4}	0.4872	1.267		
					0.5592×10^{-4}	0.4738
15	32768	0.3866×10^{-4}	0.4872	1.267		
					0.2796×10^{-4}	0.4738
16	65536	0.1933×10^{-4}	0.4872	1.267		
					0.1398×10^{-4}	0.4738
17	131072	0.9662×10^{-5}	0.4871	1.266		
					0.6990×10^{-5}	0.4738
18	262144	0.4831×10^{-5}	0.4871	1.266		
					0.3495×10^{-5}	0.4737
19	524288	0.2415×10^{-5}	0.4871	1.266		
					0.1747×10^{-5}	0.4737
20	1048576	0.1208×10^{-5}	0.4871	1.266		

Table 5-3 Window protocol performance results for $w = 2^k$

success follows a collision, assuming that the initial time interval was chosen properly, the protocol should select the next time interval to correspond to the second part of the interval divided after the collision. This is equivalent to setting W = R at step 3 by the Figure 5-1 Window protocol. Hence from these results we expect that the Figure 5-1 Window protocol, with switching points given in Section 5.2.1, has a performance which is close to the optimal Figure 3-5 Window protocol for large N.

To complete this section, we examine how the average delay D behaves as $N \to \infty$. From Gallager's infinite population results, we have that D remains bounded as $N \to \infty$ so long as the throughput $P_s < 0.4871$. For $P_s > 0.4871$, we begin by writing D as a function of P_s, N, and q. From (2.10) and (2.11) we have

$$D = 1 + N/P_s - 1/p \tag{5.24}$$

It follows from (5.7) and (5.22) that

$$p = 1 - (1-q)^{P_s/(qN)}$$

so that (5.24) becomes

$$D = 1 + N/P_s - [1 - (1-q)^{P_s/(qN)}]^{-1} \tag{5.25}$$

We have that for any given step 1 window size w, P_s is a rational function of q; thus by fixing q we hold P_s constant. Now differentiating (5.25) with respect to N and then taking the limit as $N \to \infty$, we obtain

$$\lim_{N \to \infty} \frac{\partial D}{\partial N} = \frac{1}{P_s}\left(1 + \frac{q}{\ln(1-q)}\right) \tag{5.26}$$

Hence from (5.26) we see that D increases linearly with N (in an asympotic sense) for $P_s > 0.4871$.

This result makes $P_s = 0.4871$ an interesting threshold point as we allow $N \to \infty$. For any given $P_s < 0.4871$ we have that the average delay D remains bounded, but that the average step 1 window size w increases linearly with (i.e., is some fraction of) the population size N. For any given $P_s > 0.4871$ we have that w remains fixed, but that D increases linearly with N.

Finally, using (5.7), we may write (5.26) as

$$\lim_{N \to \infty} \frac{\partial D}{\partial N} = \frac{1}{r_w}\left(\frac{1}{q} + \frac{1}{\ln(1-q)}\right) \qquad (5.27)$$

For small q, we have that

$$\left(\frac{1}{q} + \frac{1}{\ln(1-q)}\right) \approx \frac{1}{2}$$

so that (5.27) becomes

$$\lim_{N \to \infty} \frac{\partial D}{\partial N} \approx \frac{1}{2r_w} \approx \frac{T}{2N} \qquad \text{for } q \ll 1$$

which implies that $D \approx T/2$. This is intuitively pleasing since for small q, the generation time of a randomly selected packet will be uniformly distributed (in the discrete sense) over the interval of T time-slots preceding the transmission of the packet.

CHAPTER 6

CONCLUSIONS AND OPEN PROBLEMS

6.1 Conclusions

The overall objective of this thesis has been to gain a better understanding of the general multiple access problem for a finite user population. With this goal in mind, we first selected a model of the user population and communication channel which both typified the problem and yet was analytically tractable. Next, it was necessary to select measures of protocol performance for this user-channel model. After stating several common steady-state performance measures, we proceeded to show that the measures are both simply related and correspond to equivalent optimization problems.

The rest of the thesis emphasized the development and analysis of multiaccess protocols. Starting with the most general class, the Team protocols, four related classes of multiaccess protocols were defined and examined. In each case, the problems associated with determining an optimal protocol within the given class were identified. Only the last class examined, the Window protocols, proved to be tractable and so the remainder of the thesis was devoted to its analysis and characterization.

Using results from Markov decision theory, optimal Window protocols were derived for the cases of two and three users. Due to the size of the state space, an exact analysis was impractical for large user populations. Hence, a reasonable subclass of Window protocols was defined and an approximate analysis undertaken which was found to be effective in characterizing the performance and dynamic behavior of protocols within this subclass. In addition, the analysis helped to establish a link

between the finite and infinite user population problems.

6.2 Open Problems

The presentation of the results contained in this thesis leaves open several avenues for further research in the multiaccess area. These range from highly theoretical to somewhat practical in nature.

On the theoretical side, it is clearly of interest to obtain a better understanding of the class of Team protocols. The goal, of course, is to find the (or an) optimal multiaccess protocol within this, the most general class of protocols. Results obtained by Paradis [P1] indicate that the optimal Window protocol for N = 2 (as given by (4.4)) is also the optimal Team protocol. Unfortunately, the analysis does not readily carry over to larger user populations.

On a more tangible level, there is the question of the state space characterization for the Access Set and Extended Access Set protocols. Intuitively, it seems that those policies which lead to an infinite state space may be eliminated from consideration in the search for an optimal protocol. If so, then only the class of stationary deterministic policies need to be considered. Of course the state space will grow with N at least as rapidly as it does with the class of Window protocols. Hence one would encounter the same difficulties in determining an optimal policy exactly when N is large.

Promising areas for further research stem from the Window protocol analysis for large N given in Chapter 5. The presentation here sets the groundwork and indicates the type of results that are forthcoming. It is clear, however, that further work is required. In addition, it is of interest to extend the Window protocol, and this approach to its analysis,

to more practical user-channel models, such as those discussed in Section 1.3.

APPENDIX A

ON COMPUTING THE LIMITING STATE PROBABILITIES FOR MARKOV PROCESSES WITH HESSENBERG TRANSITION PROBABILITY MATRICES

In this appendix we derive a numerically stable algorithm for solving for the limiting state probabilities of a finite state, discrete-time, homogeneous Markov process with an upper Hessenberg state transition probability matrix. This structure for the state transition probability matrix is encountered in the performance analysis of perfect scheduling in Section 2.2, but more generally arises in single server systems which allow bulk arrivals to the service facility.

For notational convenience, we assume that the Markov process has N states which are identified by the integers 1 through N. The (one-step) state transition probability matrix is denoted by P, where element p_{ij} is the conditional probability of moving to state j given that the process is currently in state i. As indicated, P is an upper Hessenberg matrix which implies that all elements below the lower subdiagonal are zero (i.e., $p_{ij} = 0$ for $i > j+1$). In addition, we assume that the Markov chain associated with P is irreducible and aperiodic. Hence the limiting state probabilities π_i, $i = 1,\ldots,N$, defined by

$$\pi_i = \lim_{M \to \infty} \Pr[\text{state} = i \text{ at time } M]$$

exist, are independent of any initial state probability distribution, and are uniquely determined by the equations

$$\pi_j = \sum_{i=1}^{N} \pi_i p_{ij} \qquad j = 1,\ldots,N$$

$$1 = \sum_{i=1}^{N} \pi_i \qquad \text{(S)}$$

It is the system of equation (S) that we wish to solve numerically.

Letting π denote the row vector of limiting state probabilities, we may rewrite (S) in vector form as

$$\pi P = \pi$$

$$\pi(1,\ldots,1)^T = 1$$

where T denotes transpose. Hence it is apparent that π is the left eigenvector of P associated with the eigenvalue $\lambda = 1$, which satisfies $\pi(1,\ldots,1)^T = 1$. The method we describe computes, through an application of the Q-R algorithm for Hessenberg matrices [B1,C2], the left eigenvector q associated with $\lambda = 1$ which has norm 1 (i.e., $qq^T = 1$). Then π may be obtained by $\pi = q/q(1,\ldots,1)^T$. The essential features of the method for computing q are contained in the following proposition and its proof.

Proposition: Let $H = P - I$, where I is the identity matrix, and

$$J_i = J(i,c_i,s_i) = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & c_i & s_i & \\ & & -s_i & c_i & \\ & & & & \ddots \\ & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ i \\ i+1 \\ \\ \\ \end{matrix}$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i \quad i+1$$

where $c_i = \cos(\theta_i)$ and $s_i = \sin(\theta_i)$. Suppose that for each $i \in \{1,\ldots,N-1\}$, θ_i is chosen so that

$$\underbrace{(J_{N-1}\cdot J_{N-2}\cdots J_1)}_{Q}H = R$$

where R is an upper triangular matrix. Then the eigenvector q is equal to the last row Q.

Proof: First note that the multiplication J_kA, where A is an NxN matrix, only affects rows k and k+1 of A. Now observe that to form the upper triangular matrix R, θ_k, $k = 1,\ldots,N-1$, is chosen so that the multiplication $J_k\cdot(J_{k-1}\cdots J_1)H$ produces a 0 in the (k+1,k) position of $(J_{k-1}\cdots J_1)H$. That is,

$$\begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix}\begin{bmatrix} \tilde{h}_{kk} \\ h_{k+1,k} \end{bmatrix} = \begin{bmatrix} r_{kk} \\ 0 \end{bmatrix}$$

where $\tilde{h}_{kk}$ is the (k,k) element of $(J_{k-1}\cdots J_1)H$ and $h_{k+1,k}$ is the (k+1,k) element of this product which is also equal to the (k+1,k) element of H. Since by assumption the Markov chain corresponding to P is irreducible, we have that $h_{i+1,i} \neq 0$ for $i = 1,\ldots,N$. Hence, since θ_k is selected so that

$$s_k\tilde{h}_{kk} = c_kh_{k+1,k}$$

it follows that

$$\begin{aligned} r_{kk} &= c_k\tilde{h}_{kk} + s_kh_{k+1,k} \\ &= (\tilde{h}_{kk}^2 + h_{k+1,k}^2)^{1/2} > 0 \end{aligned}$$

for $k = 1,\ldots,N-1$. Now note that since the Markov chain corresponding to P has only one irreducible set of states, P is of rank N-1. This implies that the triangular matrix $R = QH$ is also of rank N-1, so that for some $i \in \{1,\ldots,N\}$, $r_{ii} = 0$. However, we have shown that $r_{ii} > 0$ for $i = 1,\ldots,N-1$; hence $r_{NN} = 0$. Thus we have that

$$qH = \text{last row of } R = 0$$

so that

$$qP = q$$

QED

From this result we obtain the following numerically stable algorithm for computing q:

$H \leftarrow P - I$

$q_1 \leftarrow 1$

For $k = 1,\ldots,N-1$

  $x \leftarrow h_{kk}$

  $y \leftarrow h_{k+1,k}$

  Determine c_k and s_k by Algorithm *

  For $j = k,\ldots,N$

$$\begin{bmatrix} h_{kj} \\ h_{k+1,j} \end{bmatrix} \leftarrow \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \begin{bmatrix} h_{kj} \\ h_{k+1,j} \end{bmatrix}$$

  $q_{k+1} \leftarrow c_k$

  For $j = 1,\ldots,k$

    $q_j \leftarrow -s_k q_j$

End

```
Algorithm *
   If x = 0 then
   |   c ← 1
   |   s ← 0
   Else
   |   If |x| > |y| then
   |   |   t ← y/x
   |   |   c ← 1/(1 + t^2)^1/2
   |   |   s ← ct
   |   Else
   |   |   t ← x/y
   |   |   s ← 1/(1 + t^2)^1/2
   |   |   c ← st
   Return
```

Finally, determining the limiting state probabilities for a finite-state, continuous-time Markov process involves solving the system of equations

$$\pi\tilde{Q} = 0$$

$$\pi(1,\ldots,1)^T = 1$$

where $\tilde{Q}$, the infinitesimal generator for the process, is a matrix of transition rates. If $\tilde{Q}$ is upper Hessenberg and the Markov process is irreducible, then the above results also hold for this situation by taking $H = \tilde{Q}$.

APPENDIX B

ON SELECTING AN ACCESS SET TO MAXIMIZE THE PROBABILITY OF A SUCCESSFUL TRANSMISSION

We consider the problem of selecting an access set A at the beginning of a slot to maximize the probability that the slot contains a successful transmission (i.e., contains exactly one packet). That is, we are interested in the probability of an immediate successful transmission (i.e., a horizon of one) as opposed to the steady-state probability of a success or system throughput P_s. We consider three cases. In Case 1 each user i, i = 1,...,N, independently has a packet available for transmission with probability P_i. The two other cases correspond to the same situation except that it is known in Case 2 that at least one user in the set {1,...,N} has a packet and in Case 3 that at least two users have packets.

Let $p_s(A)$, $p_s'(A)$, and $p_s''(A)$ denote the probability of a successful transmission for Cases 1, 2, and 3, respectively. For convenience we number the users in order of decreasing probability so that $P_1 \geq P_2 \geq \ldots \geq P_N$. Moreover, without loss of generality, we assume that $P_1 < 1$ and $P_N > 0$; since if $P_1 = 1$, $p_s(A)$, $p_s'(A)$, and $p_s''(A)$ are clearly maximized by setting A = {1}, and if for any user i, $P_i = 0$, whether user i is included in A or not will not affect the channel outcome in any of the three cases.

Case 1

The main results for Case 1 are stated in the form of two theorems. Both theorems are concerned with the problem:

$$\underset{A \subseteq \{1,\ldots,N\}}{\text{maximize}} \quad P_s(A) \qquad (\underline{P})$$

Theorem 1 gives the structure of a solution A^* to ($\underline{P}$), and Theorem 2 describes a simple method for obtaining this solution.

Theorem 1: A solution to problem ($\underline{P}$) is of the form $A^* = \{1,2,\ldots,k\}$ for some $k \in \{1,\ldots,N\}$ (i.e., A^* consists of all users 1 to k and does not include users k+1 to N for $k < N$).

Proof: It follows that

$$P_s(A) = \sum_{i \in A} P_i \prod_{\substack{j \in A \\ j \neq i}} (1-P_j)$$

and hence for any $i \in A$, $P_s(A)$ is linear in P_i. Now suppose that a solution A^* to ($\underline{P}$) contains at least one "gap". That is, for at least one $i \in \{1,2,\ldots,N-1\}$, $i \notin A^*$ and $i+1 \in A^*$. Since by hypothesis A^* is optimal, $P_s(A^*)$ must be a constant with respect to P_{i+1}; for if it were linearly decreasing then $P_s(A^*)$ could be increased by removing user i+1 from A^* and if it were linearly increasing then $P_s(A^*)$ could be increased by replacing user i+1 with user i. However, if $P_s(A^*)$ is a constant with respect to P_{i+1}, then the access set $\tilde{A}$ obtained by replacing user i+1 in A^* with user i is also a solution to ($\underline{P}$). Note that in obtaining the optimal access set $\tilde{A}$ from A^*, we have moved the gap from user i to user i+1. By repeating the above argument we can, maintaining an optimal solution, move the gap from

i+1 to the next higher numbered user in $\tilde{A}$, and this may be continued until this gap disappears when it is moved to user j where users j to N are not in the access set. This argument may be repeated for all remaining gaps in the original solution to (P). What we are left with is a solution to (P) of the form

$$A^* = \{1,2,\ldots,k\} \quad \text{where } k \in \{1,\ldots,N\}$$

QED

Theorem 2*: Let $K^* = \{k \mid A = \{1,\ldots,k\} \text{ solves (P)}\}$ and let

$$S(k) = \sum_{i=1}^{k} \frac{P_i}{(1-P_i)} \qquad \text{for } k = 1,\ldots,N$$

The set of optimal solutions, K^*, is given by

(1) $K^* = \{N\}$ if $S(N) \leq 1$

(2) $K^* = \{k\}$, $k \in \{1,\ldots,N\}$ if $S(N) > 1$ and $1 < S(k) < 1 + P_k/(1-P_k)$

(3) $K^* = \{k,k+1\}$, $k \in \{1,\ldots,N-1\}$ if $S(N) > 1$ and $S(k) = 1$

Proof: For convenience we define $S(0) = 0$ and we let $p_s(k) = p_s(\{1,\ldots,k\})$ where $k = 0$ denotes the empty access set so that $p_s(0) = 0$. It follows that

$$p_s(k) = \sum_{i=1}^{k} P_i \prod_{\substack{j=1 \\ j\neq i}}^{k} (1-P_j) \qquad \text{for } k = 1,\ldots,N$$

which we may write as

---

* The results of this theorem were previously stated without proof in [M5].

$$P_s(k) = S(k)\prod_{j=1}^{k}(1-P_j) \qquad \text{for } k = 0,1,\ldots,N$$

Hence we have that

$$P_s(k) - P_s(k-1) = [S(k)(1-P_k) - S(k-1)]\prod_{j=1}^{k-1}(1-P_j)$$

$$= P_k[1 - S(k-1)]\prod_{j=1}^{k-1}(1-P_j) \qquad \text{(B.1)}$$

Now since S(k) is monotonically increasing with k, and P_k and $\prod_{j=1}^{k-1}(1-P_j)$ are strictly positive for all $k \in \{1,\ldots,N\}$, from (B.1) we see that $P_s(k)$ will only be strictly increasing with k as long as $S(k-1) < 1$. Hence if $S(N) \leq 1$, then $S(N-1) < 1$ and so $K^* = \{N\}$. If $S(N) > 1$ and $1 < S(k') < 1 + P_{k'}/(1-P_{k'})$, then k' is the largest value of k such that $S(k-1) < 1$ and thus $K^* = \{k'\}$. Finally, if $S(N) > 1$ and $S(k) = 1$ for some $k \in \{1,\ldots,N-1\}$, then it follows that $P_s(k+1) = P_s(k)$ and $K^* = \{k,k+1\}$.

QED

Corollary: If $P_1 \geq 1/2$, then $p_s(A)$ is maximized by setting $A = \{1\}$.

Proof: For $P_1 \geq 1/2$ we have $S(N) \geq 1$ with equality iff $P_1 = 1/2$. Thus by Theorem 2 we have that the optimal access set is given by

$$A^* = \begin{cases} \{1\} & \text{for } P_1 > 1/2 \\ \{1\} \text{ and } \{1,2\} & \text{for } P_1 = 1/2 \end{cases}$$

Case 2

For the Case 2 problem we wish to determine an access set A which maximizes $p_s'(A)$, the conditional probability that exactly one user in A has a packet given that at least one of the N users has a packet. Letting $U = \{1,\ldots,N\}$, it follows from the definition of conditional probability that

$$p_s'(A) = \frac{p_s(A)}{(1 - p_e(U))} \tag{B.2}$$

where $p_e(U)$ is the probability that no user has a packet. Hence, since $(1 - p_e(U))$ is just a constant scaling factor, we have from (B.2) that an access set optimal for Case 1 is also optimal for Case 2 and vice versa.

Case 3

For Case 3, we wish to determine an access set A which maximizes $p_s''(A)$, the conditional probability that exactly one user in A has a packet given that at least two users in $U = \{1,\ldots,N\}$ have packets. Once more from the definition of conditional probability we have

$$p_s''(A) = \frac{p_s(A)[1 - p_e(CA)]}{p_c(U)} \tag{B.3}$$

where C denotes complement and $p_c(U)$ is the probability that at least two users have packets. The denominator term in (B.3) is again a constant scaling factor. However, note that although the term $p_s(A)$ is maximized by selecting A as given in Cases 1 and 2, this selection is not particularly good for the term $[1 - p_e(CA)]$, and hence we see the hint of a trade-off between these two terms. In fact, through a rather tedious examination of the four user case ($N = 4$) we find that, depending on the specific P_i's

satisfying $P_1 > P_2 > P_3 > P_4$, the optimal access set A^* is either {1}, {1,4}, {2,3}, {2,4}, or {3,4}. This is true even for values of P_i, $i = 1,2,3,4$, such that $A^* = \{1,2,3,4\}$ in the Case 1 problem.

APPENDIX C

ON APPLYING HOWARD'S POLICY ITERATION ALGORITHM TO THE THREE USER WINDOW PROTOCOL OPTIMIZATION PROBLEM

The optimal three user Window protocol, as specified in Table 4-3, was determined numerically using the Fortran code given below. The program utilizes Howard's policy iteration algorithm [H5] to determine, for any given $p \in (0,1)$, a policy which maximizes the system throughput P_s. The initial policy selected corresponds to that which maximizes the expected immediate reward $r_i(w_i)$, as given by (4.1), for each state S_i. Starting with this policy, each successive iteration of the algorithm finds a policy that is better until eventually no improvement in the system throughput can be made. An optimal policy for a given p was typically found in only two or three iterations, and it never took more than four.

The version of the policy iteration algorithm used within the program is based on the assumption that all policies examined by the algorithm correspond to indecomposable Markov chains. As we know from the proof for part (a) of Property 5 in Section 3.4.3, not all policies for the three user problem satisfy this condition. However, the policy given as an example in the proof, and any others corresponding to decomposable Markov chains that might exist, were never encountered by the algorithm. Typically, such a policy has a performance worse than the initially selected policy, and therefore will not be examined in any subsequent iteration.

For any given policy associated with the three user problem, each row of the state transition probability matrix will have at most two nonzero elements (in general there are at most three). Only each such nonzero

element and its position in the matrix are stored by the program. The integer variable i typically denotes the system state and k the window size. The subroutine leqt2f called during the value determination operation is an Edition 7 IMSL (International Mathematical and Statistical Library) routine which solves the system of linear equations Ax = b. The Fortran code that follows was executed on a Honeywell Level 68/DPS using a Multics operating system.

```
*  Determination of an optimal 3 user Window protocol by policy iteration
*
         real a(23,23),b(23),work(598),v(23),w(3)
         integer ss(23,3),j(23,2,3),kn(23),ko(23),idgt,ier
         double precision q(8),pe(23,3),ps(23,3),pc(23,3),z,p(23,2,3)
         character*1 char(23),c
*
*  Parameter input
         data ((ss(i,n),n=1,3),i=1,7)/1,1,1, 2,1,1, 2,2,1, 3,2,1,
        & 4,2,1, 4,3,1, 5,2,1/
         data ((ss(i,n),n=1,3),i=8,16)/1,1,1, 2,1,1, 2,2,1, 3,2,1,
        & 4,2,1, 4,3,1, 5,2,1, 8,8,2, 8,8,3/
         data ((ss(i,n),n=1,3),i=17,23)/1,1,1, 2,1,1, 3,1,1, 8,2,1,
        & 8,3,1, 8,3,2, 8,4,1/
         data (((j(i,m,k),k=1,3),m=1,2),i=1,7)/3,2,1, 3,15,8, 3,2,1,
        & 3,15,9, 4,2,1, 4,15,10, 4,2,1, 4,15,11, 4,2,1, 4,15,12,
        & 5,2,1, 5,15,13, 4,2,1, 4,15,14/
         data (((j(i,m,k),k=1,2),m=1,2),i=8,16)/17,20, 15,16, 17,20,
        & 15,16, 18,20, 15,16, 18,20, 15,16, 18,20, 15,16, 19,20,
        & 15,16, 18,20, 15,16, 21,0, 21,0, 23,0, 23,0/
         data (((j(i,m,k),k=1,2),m=1,2),i=17,23)/5,4, 22,15, 5,4,
        & 22,15, 5,4, 22,15, 4,0, 4,0, 5,0, 5,0, 6,0, 6,0, 7,0, 7,0/
         data (kn(i),i=15,16)/1,1/
         data (kn(i),i=20,23)/1,1,1,1/
         idgt=0
*
*  Computes pe(i,k), ps(i,k), and pc(i,k)
  210    write(0,1)
    1    format(//' p='$)
         input, prob
         if(prob.eq.0.) stop
         q(1)=1.-prob
         do 50 i=2,5
   50    q(i)=q(i-1)*q(1)
         q(8)=0.
         do 60 i=1,7
         pe(i,1)=q(ss(i,1))
         ps(i,1)=1.-pe(i,1)
```

```
      pc(i,1)=0.
      pe(i,2)=pe(i,1)*q(ss(i,2))
      pc(i,2)=(1.-q(ss(i,1)))*(1.-q(ss(i,2)))
      ps(i,2)=1.-pe(i,2)-pc(i,2)
      pe(i,3)=pe(i,2)*q(ss(i,3))
      pc(i,3)=pc(i,2)+ps(i,2)*(1.-q(ss(i,3)))
   60 ps(i,3)=1.-pe(i,3)-pc(i,3)
      do 70 i=8,14
      m=i-7
      pe(i,1)=q(ss(i,1))*(1.-q(ss(i,2)))*(1.-q(ss(i,3)))/pc(m,3)
      ps(i,1)=1.-pe(i,1)
      pc(i,2)=pc(m,2)/pc(m,3)
   70 ps(i,2)=1.-pc(i,2)
      do 75 i=15,16
   75 ps(i,1)=1.
      do 80 i=17,19
      z=1.-q(ss(i,1)+ss(i,2))
      ps(i,1)=(1.-q(ss(i,1)))/z
      pe(i,1)=1.-ps(i,1)
      pc(i,2)=ps(i,1)*(1.-q(ss(i,2)))
   80 ps(i,2)=1.-pc(i,2)
      do 85 i=20,23
   85 ps(i,1)=1.
*
*  Computes transition probabilities p(i,j,k)
      do 100 i=1,7
      do 100 k=1,3
      p(i,2,k)=pc(i,k)
  100 p(i,1,k)=1.-p(i,2,k)
      do 102 i=8,14
      do 102 k=1,2
      p(i,1,k)=ps(i,k)
  102 p(i,2,k)=1.-p(i,1,k)
      do 104 i=15,16
      p(i,1,1)=1.
  104 p(i,2,1)=0.
      do 106 i=17,19
      do 106 k=1,2
      p(i,1,k)=ps(i,k)
  106 p(i,2,k)=1.-p(i,1,k)
      do 108 i=20,23
      p(i,1,1)=1.
  108 p(i,2,1)=0.
*
*  Initializes variables for policy iteration algorithm
      do 110 i=1,23
      v(i)=0.
      ko(i)=0
  110 char(i)=' '
      nx=0
      write(0,2) (i,i=1,23)
    2 format(/'  n ',23i3,5x,'ps',6x,'D'/)
*
*  Policy improvement routine
```

```
      230 continue
          do 120 i=1,7
          do 130 k=1,3
      130 w(k)=ps(i,k)+ p(i,1,k)*v(j(i,1,k))+p(i,2,k)*v(j(i,2,k))
          x=w(1)
          kn(i)=1
          do 140 k=2,3
          d=w(k)-x
          if(d.gt.0.) x=w(k)
      140 if(d.gt.0.) kn(i)=k
          if(kn(i).eq.2.and.w(2).eq.w(3)) char(i)='a'
          if(kn(i).ne.1) go to 120
          if(w(1).eq.w(2)) char(i)='a'
          if(w(1).eq.w(3)) char(i)='b'
          if(w(1).eq.w(2).and.w(1).eq.w(3)) char(i)='c'
      120 continue
          do 160 i=8,14
          do 165 k=1,2
      165 w(k)=ps(i,k)+p(i,1,k)*v(j(i,1,k))+p(i,2,k)*v(j(i,2,k))
          kn(i)=1
          d=w(2)-w(1)
          if(d.gt.0.) kn(i)=2
          if(d.eq.0.) char(i)='a'
      160 continue
          do 170 i=17,19
          do 175 k=1,2
      175 w(k)=ps(i,k)+p(i,1,k)*v(j(i,1,k))+p(i,2,k)*v(j(i,2,k))
          kn(i)=1
          d=w(2)-w(1)
          if(d.gt.0.) kn(i)=2
          if(d.eq.0.) char(i)='a'
      170 continue
          isum=0
          do 180 i=1,23
      180 isum=isum+abs(kn(i)-ko(i))
          if(isum.eq.0.and.nx.eq.1) go to 210
          if(isum.eq.0.and.nx.ne.1) go to 270
          nx=nx+1
*
*  Value determination operation
          do 190 i=1,23
          do 190 m=1,23
      190 a(i,m)=0.
          do 200 i=1,23
          do 200 m=1,2
      200 a(i,j(i,m,kn(i)))=a(i,j(i,m,kn(i)))-p(i,m,kn(i))
          do 220 i=1,23
          a(i,i)=a(i,i)+1.
          a(i,23)=1.
      220 b(i)=ps(i,kn(i))
          call leqt2f(a,1,23,23,b,idgt,work,ier)
          c=' '
          if(ier.eq.129) go to 260
          if(ier.eq.131) c='*'
```

```
      do 240 i=1,22
  240 v(i)=b(i)
      v(23)=0.
      if(nx.eq.1) go to 270
      if(nx.eq.20) go to 290
  280 do 250 i=1,23
      char(i)=' '
  250 ko(i)=kn(i)
      go to 230
  260 write(0,5)
    5 format(6x,'error in leqt2f:  matrix is algorithmically singular')
      go to 210
  290 write(0,4)
    4 format(5x,'n > 20')
      go to 210
*
*  Prints results
  270 g=b(23)
      D=1.+(3./g)-(1./prob)
      write(0,3) nx,c,(kn(i),char(i),i=1,23),g,D
    3 format(i3,a1,1x,23(i2,a1),2f8.5)
      if(nx.eq.1) go to 280
      go to 210
      end
```

APPENDIX D

DERIVATION OF RECURRENCE RELATIONS FOR E[u|w] and E[t|w]

In this appendix we derive Equations (5.8) and (5.9) which may be used to determine in a recursive manner E[u|w] and E[t|w], the expected number of users processed in and the expected duration of a CRP with step 1 window size w, respectively. We begin by defining the following quantities needed for the derivation:

w' = window size for step 2 following a collision at step 1
$< w$

w'' = window size for step 3 following a success at step 2
$= w - w'$

$e(w)$ = Pr[an empty slot|w users in window with step 1 statistics]
$= (1-q)^w$

$s(w)$ = Pr[a success|w users in window with step 1 statistics]
$= wq(1-q)^{w-1}$

$c(w)$ = Pr[a collision|w users in window with step 1 statistics]

$E_c[u|w]$ = E[number of users processed in remainder of CRP| w users in R of which ≥ 2 have packets]

$E_c[t|w]$ = E[time remaining in CRP|w users in R of which ≥ 2 have packets]

Shown in Figure D-1 is the probability tree from which (5.8) and (5.9) are derived. The events of interest are boxed and to the right of each box is the probability associated with that point on the tree. To make the

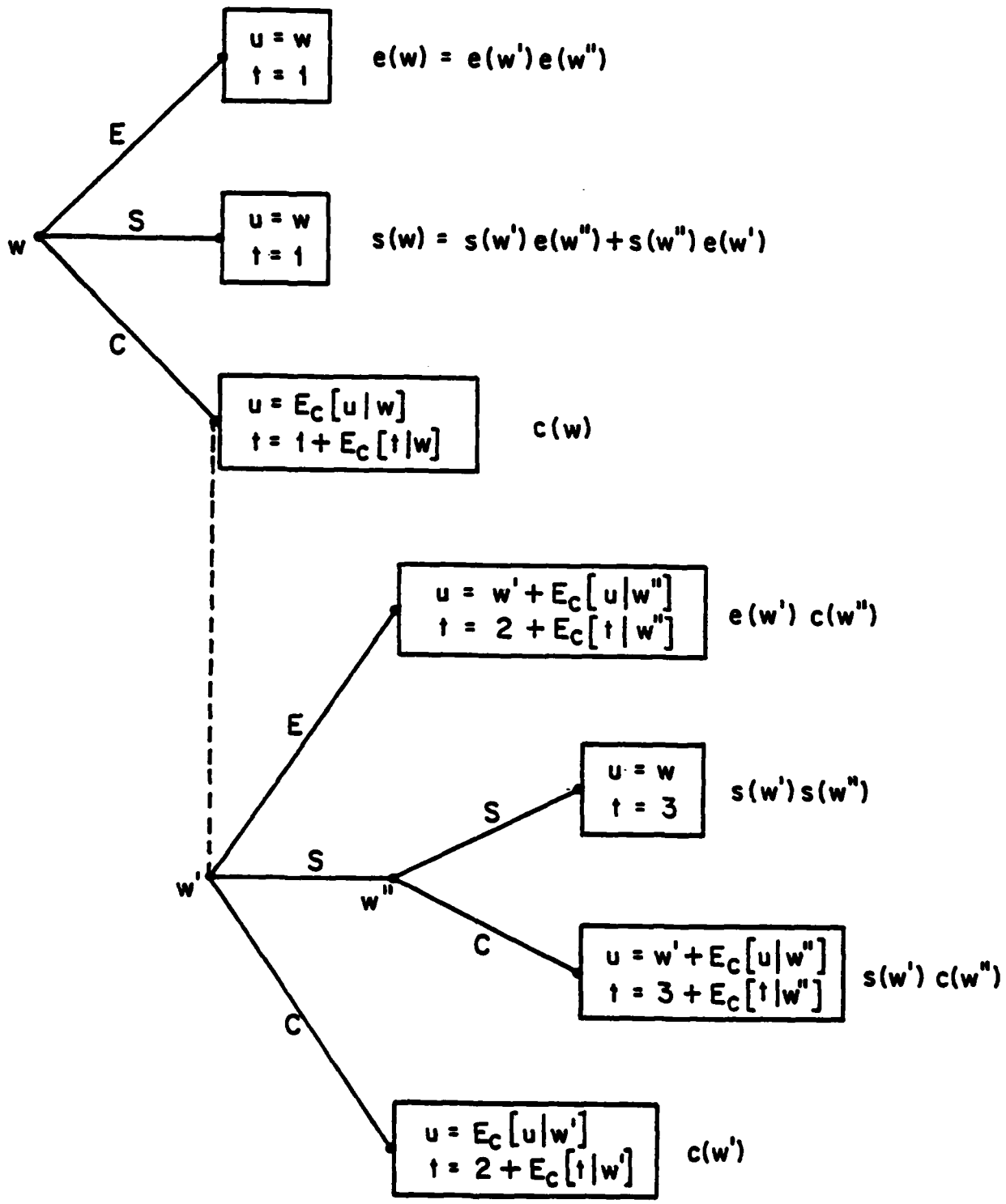


Figure D-1 Probability tree for E[u|w] and E[t|w] derivation

equations that follow less cumbersome, we adopt the simplifying notation $e = e(w)$, $e' = e(w')$, and $e'' = e(w'')$ and likewise for s and c. From the first branching point on the tree we have

$$E[u|w] = w(e + s) + E_c[u|w]c$$
$$E[t|w] = 1 + E_c[t|u]c$$

from which we obtain

$$E_c[u|w] = \frac{E[u|w] - w(e + s)}{c} \tag{D.1}$$

$$E_c[t|w] = \frac{E[t|w] - 1}{c} \tag{D.2}$$

Now from the leaves of the Figure D-1 probability tree we have

$$\begin{aligned} E[u|w] &= w(e'e'' + s'e'' + s''e' + s's'') \\ &\quad + (w' + E_c[u|w''])(e' + s')c'' + E_c[u|w']c' \\ &= (w - w')(e' + s')(e'' + s'') + w'(e' + s') \\ &\quad + E_c[u|w''](e' + s')c'' + E_c[u|w']c' \end{aligned}$$

and using (D.1) we obtain

$$E[u|w] = E[u|w'] + E[u|w''](e' + s') \tag{D.3}$$

Similarly, we have that

$$\begin{aligned} E[t|w] &= 1 + (1 + E_c[t|w''])e'c'' \\ &\quad + 2s's'' + (2 + E_c[t|w''])s'c'' + (1 + E_c[t|w'])c' \\ &= 1 + (e'c'' + 2s's'' + 2s'c'' + c') \\ &\quad + E_c[t|w''](e' + s')c'' + E_c[t|w']c' \end{aligned}$$

and using (D.2) we obtain after some manipulation

$$E[t|w] = 1 - e'(1 + s'') - e''(e' + 2s') + E[t|w'] + E[t|w''](e' + s') \qquad (D.4)$$

Equations (D.3) and (D.4) correspond to, respectively, (5.8) and (5.9).

REFERENCES

[A1] N. Abramson, "The ALOHA System - Another Alternative for Computer Communications," AFIPS Conf. Proc., vol. 37, pp. 281-285, 1970.

[A2] K. Astrom, "Optimal Control of Markov Processes with Incomplete State Information," J. Math. Anal. Appl., vol. 10, pp. 174-205, 1965.

[B1] R. Bartels, G. Golub, C. Van Loan, Applied Matrix Computations, to be published by Johns Hopkins Press, Baltimore, MD.

[B2] D. Bertsekas, Dynamic Programming and Stochastic Control, Academic Press, New York, NY, 1976.

[C1] J. Capetanakis, "Tree Algorithms for Packet Broadcast Channels," IEEE Trans. on Inform. Theory, vol. IT-25, pp. 505-515, Sept. 1979.

[C2] J. Capetanakis, "Generalized TDMA: The Multi-Accessing Tree Protocol," IEEE Trans. on Comm., vol. COM-27, pp. 1476-1484, October 1979.

[C3] A. Carleial, M. Hellman, "Bistable Behavior of ALOHA-Type Systems," IEEE Trans on Comm., vol. COM-23, pp. 401-409, April 1975.

[C4] I. Chlamtac, W. Franta, K. Levin, "BRAM: The Broadcast Recognizing Access Method," IEEE Trans. on Comm., vol. COM-27, pp. 1183-1190, August 1979.

[C5] W. Crowther, R. Rettbert, D. Walden, S. Ornstein, F. Heart, "A System for Broadcast Communication: Reservation-ALOHA," Proc. of Sixth Hawaii Int. Sys. Sci. Conf., Jan. 1973.

[C6] R. Cruz, B. Hajek, "A New Upper Bound to the Throughput of a Multi-Access Broadcast Channel," Electrical Engineering Dept., Univ. of Illinois, Urbana, IL, 1980.

[D1] M. DeGroot, Optimal Statistical Decisions, McGraw-Hill, New York, NY, 1970.

[D2] C. Derman, Finite State Markovian Decision Processes, Academic Press, New York, NY, 1970.

[F1] G. Fayolle, E. Gelenbe, J. Labetoulle, "Stability and Optimal Control of the Packet Switching Broadcast Channel," Jour. of ACM, vol. 24, pp. 375-386, July 1977.

[F2] W. Feller, An Introduction to Probability Theory and its Applications, Vol. I, John Wiley & Sons, New York, NY, 1968.

[G1] R. Gallager, "Conflict Resolution in Random Access Broadcast Networks," Proc. of AFOSR Workshop on Comm. Theory and Appl., pp. 74-76, Sept. 1978.

[G2] A. Gourlay, G. Watson, Computational Methods for Matrix Eigen-problems, John Wiley & Sons, New York, NY, 1973.

[H1] B. Hajek, T. van Loon, "Decentralized Dynamic Control of a Multi-Access Broadcast Channel," Electrical Engineering Dept., Univ. of Illinois, Urbana, IL, 1980.

[H2] L. Hansen, M. Schwartz, "An Assigned-Slot Listen-Before-Transmission Protocol for a Multiaccess Data Channel," IEEE Trans. on Comm., vol. COM-27, pp. 846-857, June 1979.

[H3] J. Hayes, "An Adaptive Technique for Local Distribution," IEEE Trans. on Comm., vol. COM-26, pp. 1178-1186, Aug. 1978.

[H4] Y.C. Ho, K.C. Chu, "Team Decision Theory and Information Structures in Optimal Control Problems - Part I," IEEE Trans. on Automatic Control, vol. AC-17, pp. 15-22, Feb. 1972.

[H5] R. Howard, Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA, 1960.

[H6] P. Humblet, "Bounds on the Utilization of Aloha-like Multiple Access Broadcast Channels," Report LIDS-P-1000, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, June 1980.

[H7] P. Humblet, "Data Communication Networks and Information Theory," NTC Conf. Rec., pp. 20.3.1-20.3.5, Nov. 1980.

[H8] P. Humblet, J. Mosely, "Efficient Accessing of a Multiaccess Channel," Proc. of IEEE Conf. on Dec. & Control, pp. 624-627, Dec. 1980.

[J1] I. Jacobs, R. Binder, E. Hoversten, "General Purpose Packet Satellite Networks," Proc. IEEE, vol. 66, Nov. 1978.

[K1] R. Kahn, S. Gronemeyer, J. Burchfiel, R. Kunzelman, "Advances in Packet Radio Technology," Proc. of the IEEE, vol. 66, pp. 1468-1496, Nov. 1978.

[K2] L. Kleinrock, S. Lam, "Packet-Switching in a Slotted Satellite Channel," AFIPS Conf. Proc., vol. 42, pp. 703-710, June 1973.

[K3] L. Kleinrock, S. Lam, "Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation," IEEE Trans. on Comm., vol. COM-23, pp. 410-423, April 1975.

[K4] L. Kleinrock, F. Tobagi, "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics," IEEE Trans. on Comm., vol. COM-23, pp. 1400-1416, Dec. 1975.

[K5] L. Kleinrock, Queueing Systems, Vol. 1: Theory, Wiley-Interscience, New York, NY, 1975.

[K6] L. Kleinrock, M. Scholl, "Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes for a Small Number of Data Users," ICC Conf. Proc., pp. 22.1-105 - 22.1-111, June 1977.

[K7] L. Kleinrock, Y. Yemini, "An Optimal Adaptive Scheme for Multiple Access Broadcast Communication," ICC Conf. Proc., pp. 7.2.1-7.2.5, June 1978.

[L1] S. Lam, L, Kleinrock, "Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures," IEEE Trans. on Comm., vol. COM-23, Sept. 1975.

[L2] S. Lam, "Satellite Packet Communication - Multiple Access Protocols and Performance," IEEE Trans. on Comm., vol. COM-27, pp. 1456-1466, October 1979.

[M1] J. Marschak, "Elements for a Theory of Teams," Management Science, vol. 1, pp. 127-137, 1955.

[M2] J. Marschak, R. Radner, Economic Theory of Teams, Yale Univ. Press, New Haven, CT, 1972.

[M3] J. Martin, Teleprocessing Network Organization, Prentice-Hall, Englewood Cliffs, NJ, 1970.

[M4] R. Metcalfe, D. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Comm. of the ACM, vol. 19, pp. 395-404, July 1976.

[M5] C. Meubus, M. Kaplan, "Protocols for Multi-Access Packet Satellite Communications," NTC Conf. Rec., pp. 11.4.1-11.4.7, Dec. 1979.

[M6] K. Mittal, A. Venetsanopoulos, "On the Dynamic Control of the Urn Scheme for Multiple Access Broadcast Communication Systems," IEEE Trans. on Comm., vol. COM-29, pp. 962-970, July 1981.

[M7] M. Molle, "On the Capacity of Infinite Population Multiple Access Protocols," Computer Science Dept., UCLA, Los Angeles, CA, March 1980.

[M8] J. Mosely, "An Efficient Contention Resolution Algorithm for Multiple Access Channels," Report LIDS-TH-918, Laborotory for Information and Decision Systems, MIT, Cambridge, MA, May 1979.

[P1] A. Paradis, "Application of Optimal Control to the Multiple Access Channel," S.M. Thesis, Dept. of Elec. Eng. and Comp. Sci., MIT, Cambridge, MA, June 1981.

[P2] N. Pippenger, "Bounds on the Performance of Protocols for a Multiple Access Broadcast Channel," Report RC-7742, Math. Science Dept., IBM Thomas J. Watson Research Center, Yorktown Heights, NY, June 1979.

[R1] G. Ricart, A. Agrawala, "Dynamic Management of Packet Radio Slots," Third Berkeley Workshop on Distributed Data Mang. and Computer Networks, August 1978.

[R2] L. Roberts, "Aloha Packet System with and without Slots and Capture," ASS Note 8, June 1972; reprinted in Computer Comm. Rev., vol. 5, pp. 28-42, April 1975.

[R3] L. Roberts, "Dynamic Allocation of Satellite Capacity Through Packet Reservation," AFIPS Conf. Proc., vol. 42, pp. 711-716, June 1973.

[R4] S. Ross, Applied Probability Models with Optimization Applications, Holden-Day, San Francisco, CA, 1970.

[S1] R. Smallwood, E. Sondik, "The Optimal Control of Partially Observable Markov Processes over a Finite Horizon," Oper. Res., vol. 21, pp. 1071-1088, 1973.

[S2] M. Sobel, P. Groll, "Group Testing to Eliminate Efficiently All Defectives in a Binomial Sample," Bell System Tech. Journal, vol. 38, pp. 1179-1252, 1959.

[T1] F. Tobagi, "Multiaccess Protocols in Packet Communication Systems," IEEE Trans. on Comm., vol. COM-28, pp. 468-488, April 1980.

[T2] D. Towsley, J. Wolf, "An Application of Group Testing to the Design of Multi-User Access Protocols," Dept. of Elec. and Comp. Eng., Univ. of Mass., Amherst, MA, October 1981.

[T3] B. Tsybakov, U. Mikhailov, "An Upper Bound for Maximum Throughput of Random Access Systems," Institute for Problems of Information Transmission, Moscow, USSR, 1981.